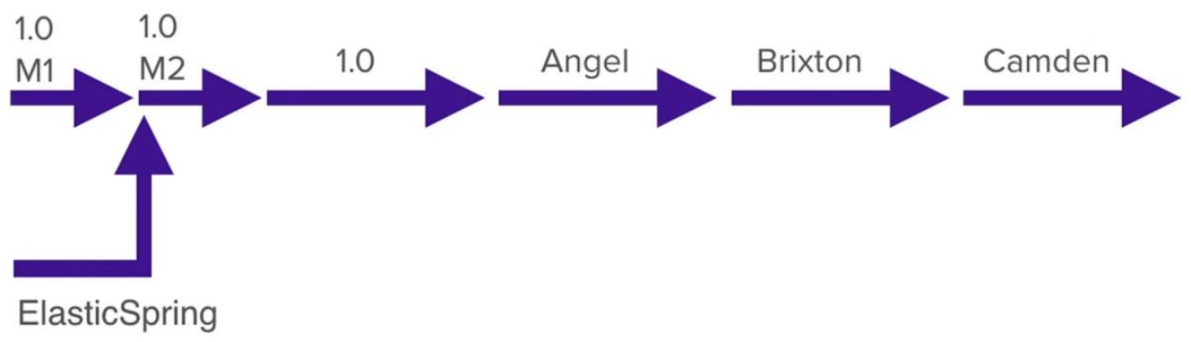
Spring Cloud on AWS

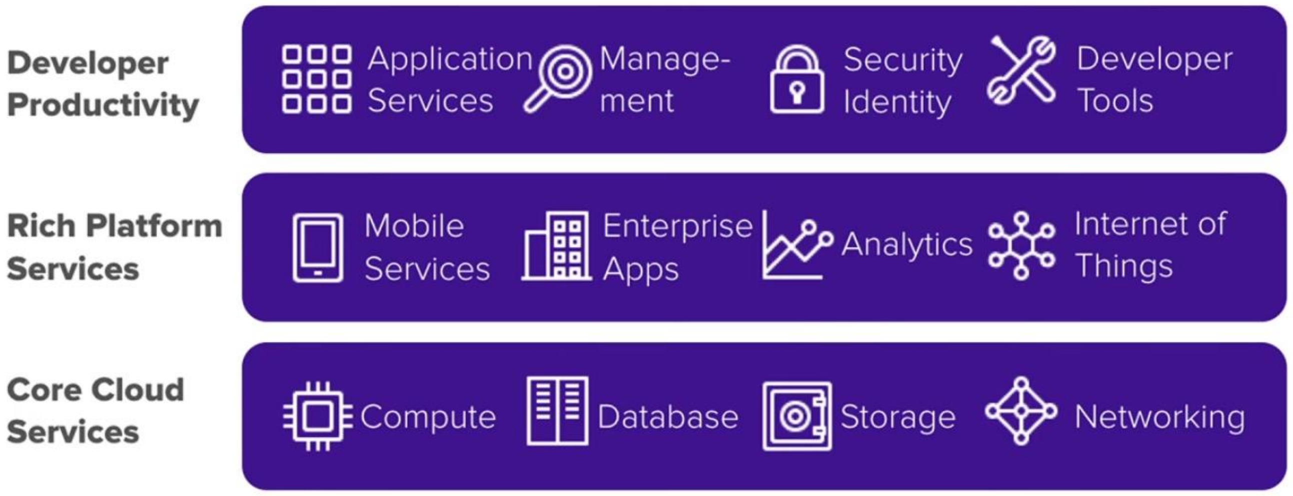
Agim Emruli
@aemruli

Pivotal

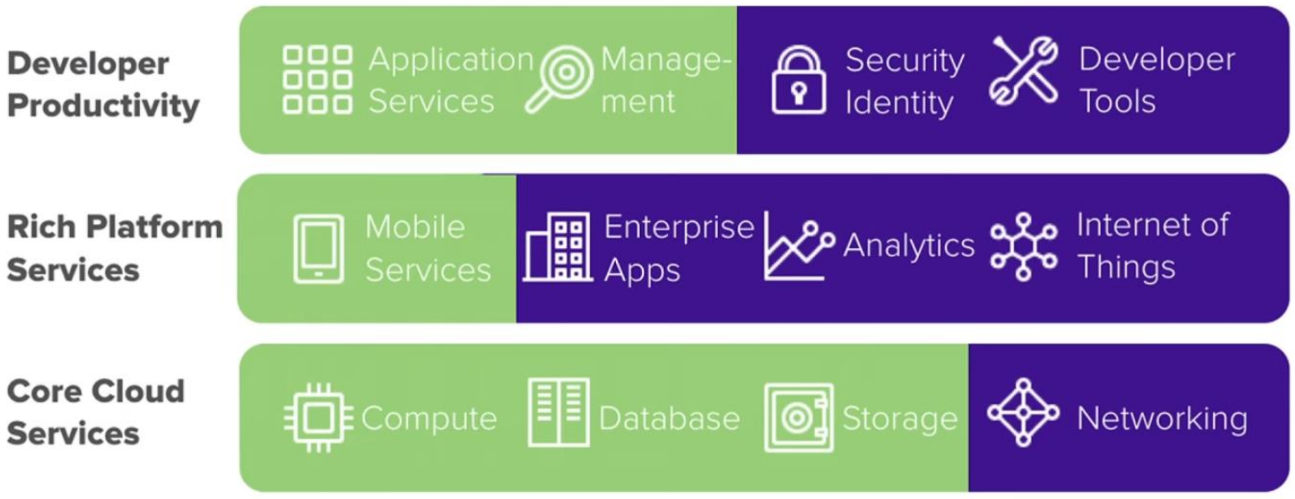
Spring Cloud for AWS - History



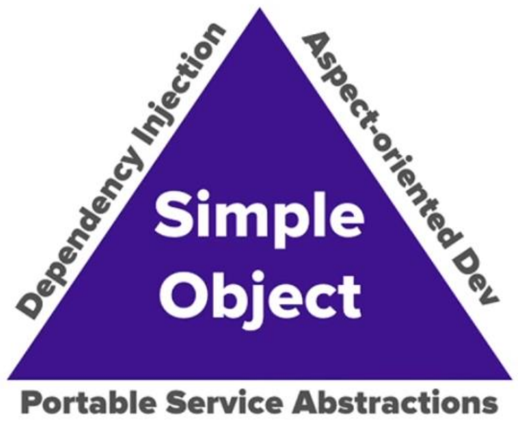
Amazon Web Services



Spring Cloud for AWS

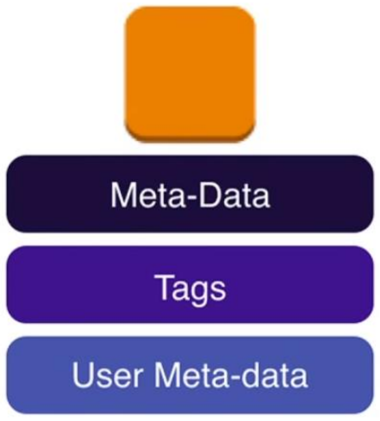


Spring Triangle



Dependency Injection with Environment

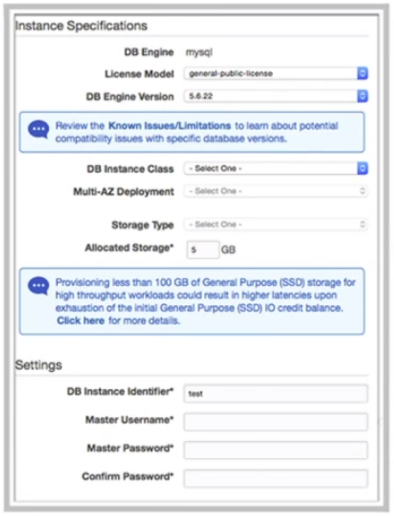
```
@Component
public class ApplicationInfoBean {

    @Value("${ami-id}")
    private String amiId;

    @Value("${hostname}")
    private String hostname;

    @Value("${instance-type}")
    private String instanceType;

    @Value("${services/domain}")
    private String serviceDomain;
}
```



When we run in the cloud, we want to know where we are running and what we are running. We can inject metadata into our apps and use them on AWS.

Dependency Injection with Services

```
@EnableRdsInstance(
   dbInstanceIdentifier = "test",
   password = "${user.password}",
   readReplicaSupport = true)
public static class AppConfig {}
```



Instance Specifications

DB Engine	mysql
License Model	general-public-license
DB Engine Version	5.6.22

Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class	- Select One -
Multi-AZ Deployment	- Select One -
Storage Type	- Select One -
Allocated Storage*	5 GB

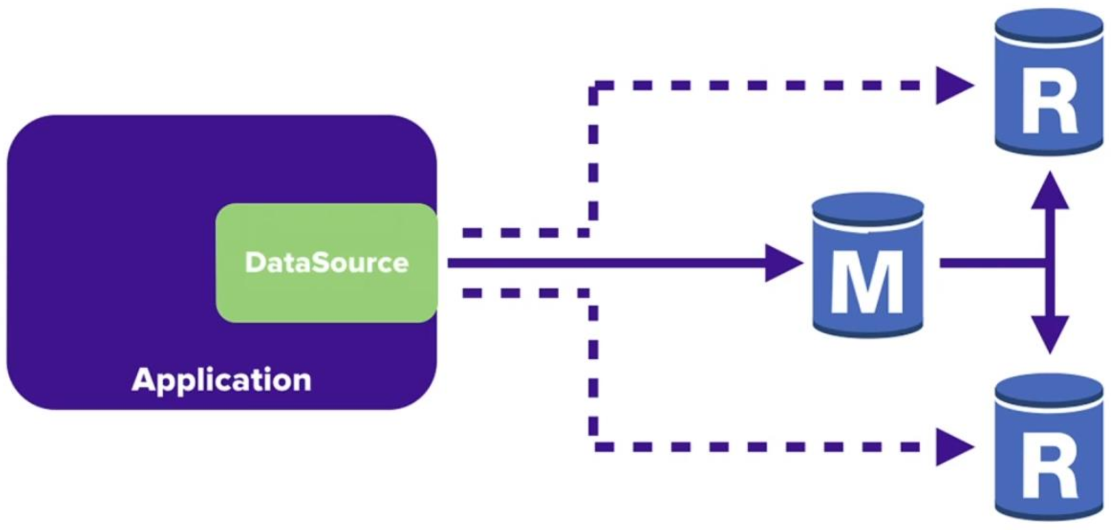
Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. **Click here** for more details.

Settings

DB Instance Identifier*	test
Master Username*	
Master Password*	
Confirm Password*	

Don’t write code to retrieve the instance details of your RDS instance, use an annotation instead via injection of the service

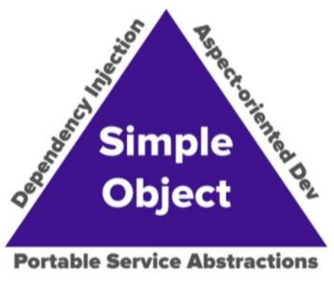
Amazon RDS Read-Replicas



Read-replica Demarcation

```
@Service
public class JdbcPersonService implements PersonService {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public JdbcPersonService(DataSource dataSource) {
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }

    @Transactional(readOnly = true)
    public List<Person> all() {
        return jdbcTemplate.query("SELECT * FROM Person", … );
    }

    @Transactional
    public void store(Person person) {
        jdbcTemplate.update("INSERT INTO Person …");
    }
}
```

Spring Triangle



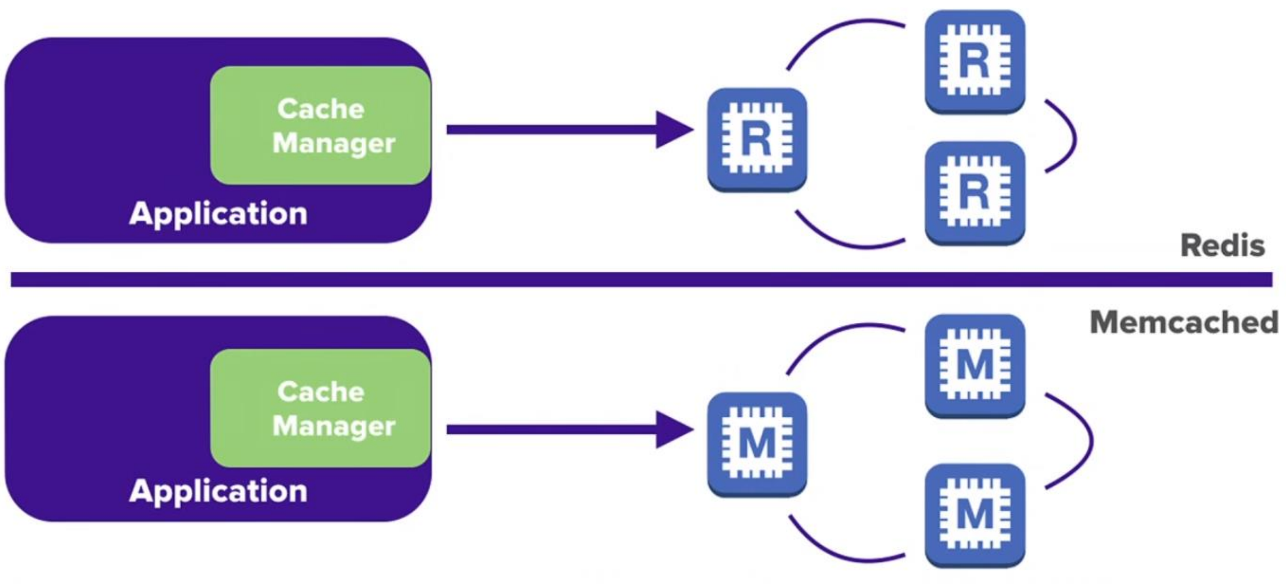
Simple Storage - Resource Loader Abstraction

```
public class S3LoadingClass {

    @Autowired
    private ResourceLoader rl;

    public void download() {
        Resource resource = rl.getResource("s3://myBucket/myFile.txt");
      resource.getInputStream();
    }

    public void upload(){
        WritableResource writableResource = (WritableResource)
                rl.getResource("s3://myBucket/newFile.txt");
        writableResource.getOutputStream();
    }
}
```

Simple E-Mail Service - Mail Sender

```
public class MailSender {

    @Autowired
    private MailSender mailSender;

    @Test
    public void sendMail() throws Exception {
        SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
        simpleMailMessage.setFrom(“sender@mail.com”);
        simpleMailMessage.setTo(“recipient@mail.com”);
        simpleMailMessage.setSubject("test subject");
        simpleMailMessage.setText("test content");
        this.mailSender.send(simpleMailMessage);
    }
}
```

Elasticache - Caching



Caching Service Abstraction

```
@Service
@EnableElastiCache(
   @CacheClusterConfig(name = “myCache”))
public class ExpensiveService {

  @Cacheable("myCache")
  public String calculateExpensiveValue(String key){
    return …;
  }
}
```

Amazon Simple Queueing Service

- HTTP-based messaging service
- Only String payloads
- Pay-per message (millions)
- No transactions
- Visibility rules

Sending Messages with Spring Messaging

```
@Service
@EnableSqs
public class MessageSendingBean {

    private final QueueMessagingTemplate messagingTemplate;

    @Autowired
    public MessageSendingBean(AmazonSQS amazonSqs) {
        this.messagingTemplate = new QueueMessagingTemplate(amazonSqs);
    }

    public String sendAndReceive(String payload) {
        this.messagingTemplate.convertAndSend("requestQueue", payload);
        return this.messagingTemplate.
                        receiveAndConvert("resultQueue", String.class);
    }
  }
```

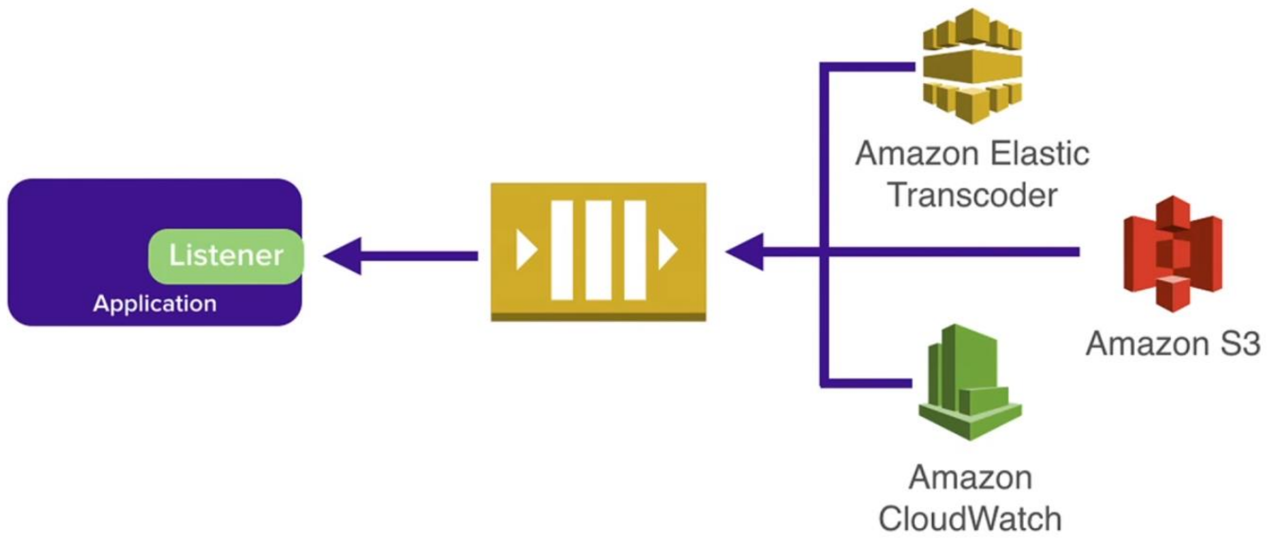
Polling Message Using Container

```
@Component
@EnableSqs
public class MessageReceivingBean {

    @SqsListener("receivingQueue")
    public @SendTo("responseQueue") Confirmation
            processEvent(CustomEvent customEvent){
        return new Confirmation();
    }
}
```

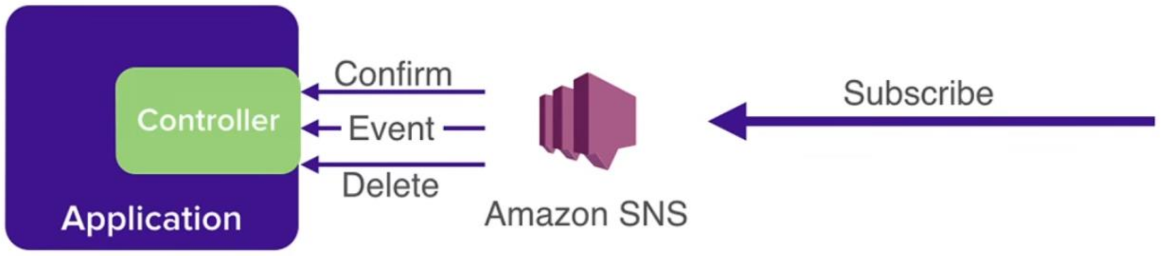
Acknowledgment

```
@Component
@EnableSqs
public class MessageReceivingBean {

    @SqsListener(value = "receivingQueue",
                 deletionPolicy = NEVER)
    public @SendTo("responseQueue") Confirmation processEvent(
         CustomEvent customEvent,
         Acknowledgment acknowledgment){

        if(successful){
           acknowledgment.acknowledge();
        }
        return new Confirmation();
    }
}
```

Combining AWS Services



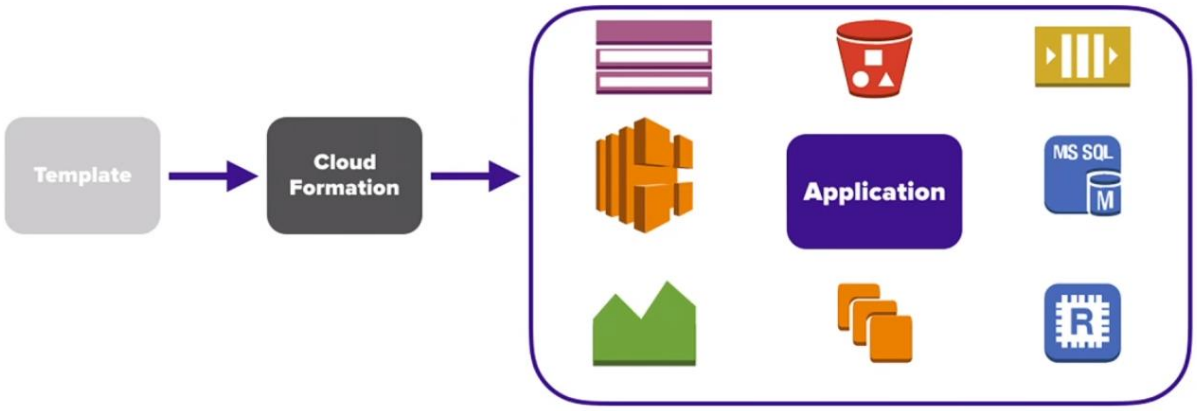
HTTP-based Notifications with Amazon SNS



Spring MVC - SNS Controller

```java
@RestController
@RequestMapping("/sns/receive")
public class SnsEndpointController {

    @NotificationSubscriptionMapping
    public void confirmSubscription(
      NotificationStatus notificationStatus) {
        notificationStatus.confirmSubscription();
    }

    @NotificationMessageMapping
    public void receiveNotification(
         @NotificationMessage String message,
         @NotificationSubject String subject) {
    }
}
```

Deployment using Cloud Formation



Cloud Formation Auto Configuration

spring-cloud-aws-autoconfigure
Application

```xml
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-aws-autoconfigure</artifactId>
</dependency>
```

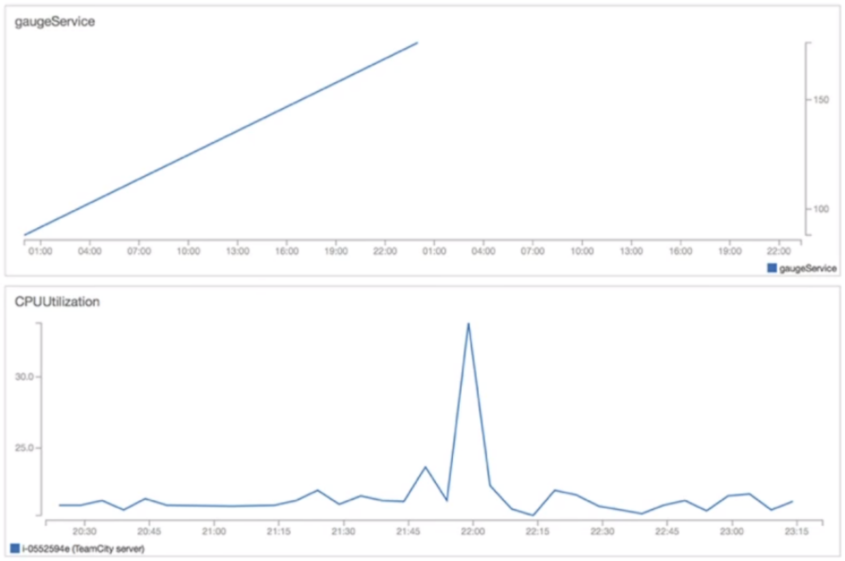
Auto Config Services

```java
@Service
public class ApplicationBean {

    @Autowired
    private DataSource dataSource;

    @Autowired
    private MailSender mailSender;

    @Cacheable("myCache")
    public void cache(){
    }

    @SqsListener("myQueue")
    public void receiveMessage(){
    }
}
```

Pushing Metrics with Spring Boot Actuator

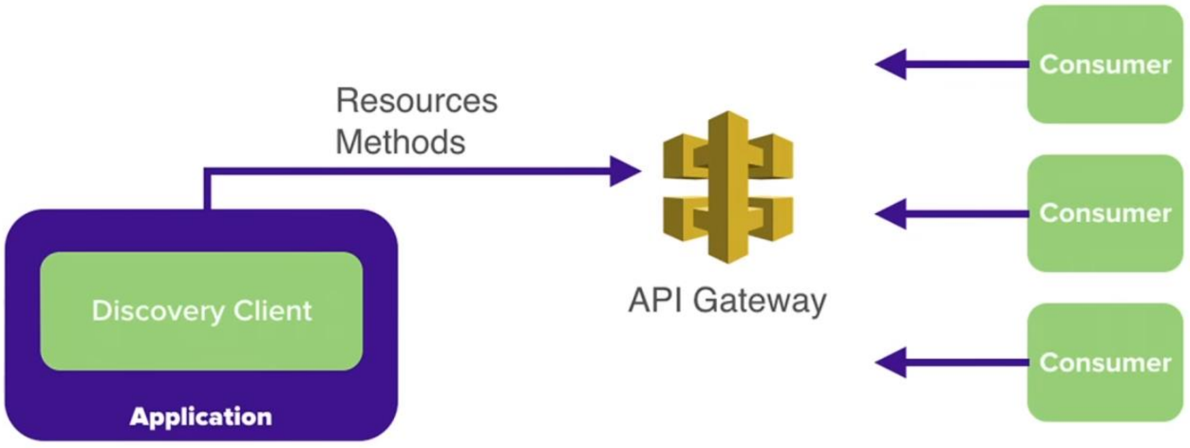
```java
@Service
public class MetricProducer {

    @Autowired
    private CounterService counterService;

    public void sendOrder() {
        this.counterService.increment("orders");
    }

    public void cancelOrder(){
        this.counterService.decrement("orders");
    }
}
```

Cloud Watch Visualization



The Future

Spring Cloud Discovery Client



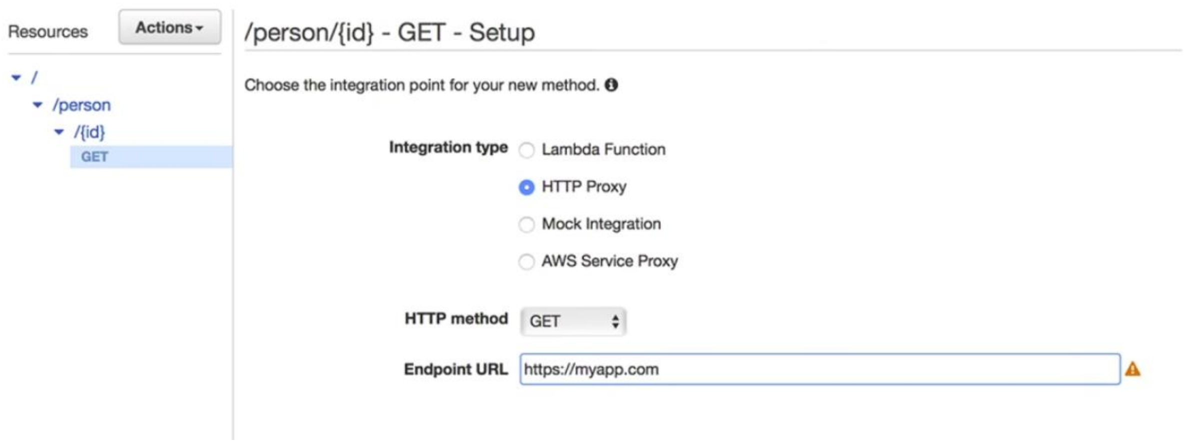
Spring Cloud Application

```
@SpringCloudApplication
public class SampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SampleApplication.class, args);
    }
}
```

Spring Cloud Controller

```
@Controller
public class SampleController {

    @RequestMapping("/person/{id}")
    public Person getPerson(@PathVariable String id) {
        return new Person("Agim", "Emruli");
    }
}
```

API Gateway Declaration



/person/{id} - GET - Setup

Choose the integration point for your new method.

Integration type ○ Lambda Function [✓ HTTP Proxy] ○ Mock Integration ○ AWS Service Proxy

HTTP method: GET

Endpoint URL: https://myapp.com

Consuming API Gateway Services

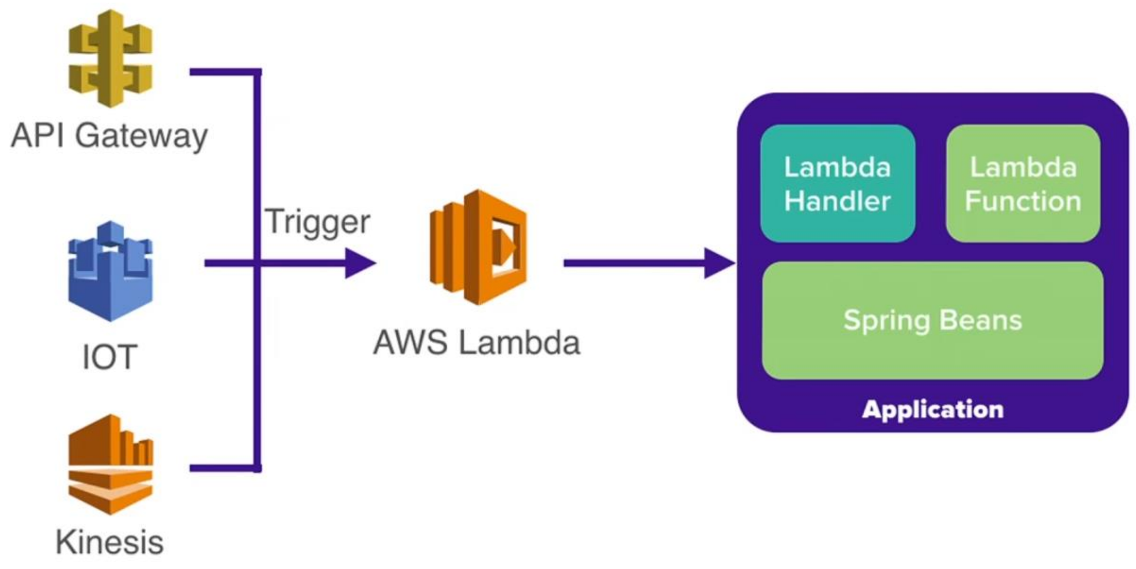
```java
@Service
public class RestMoneyExchangeGateway implements MoneyExchangeGateway {

    @Autowired
    @LoadBalanced
    private RestOperations restTemplate;

    public Double exchangeMoney(String currency, long amount) {
        return restTemplate.getForObject(“http://MY-SERVICE/exchange/{currency}/{price}”,
                           Double.class, currency, amount);
    }
}
```

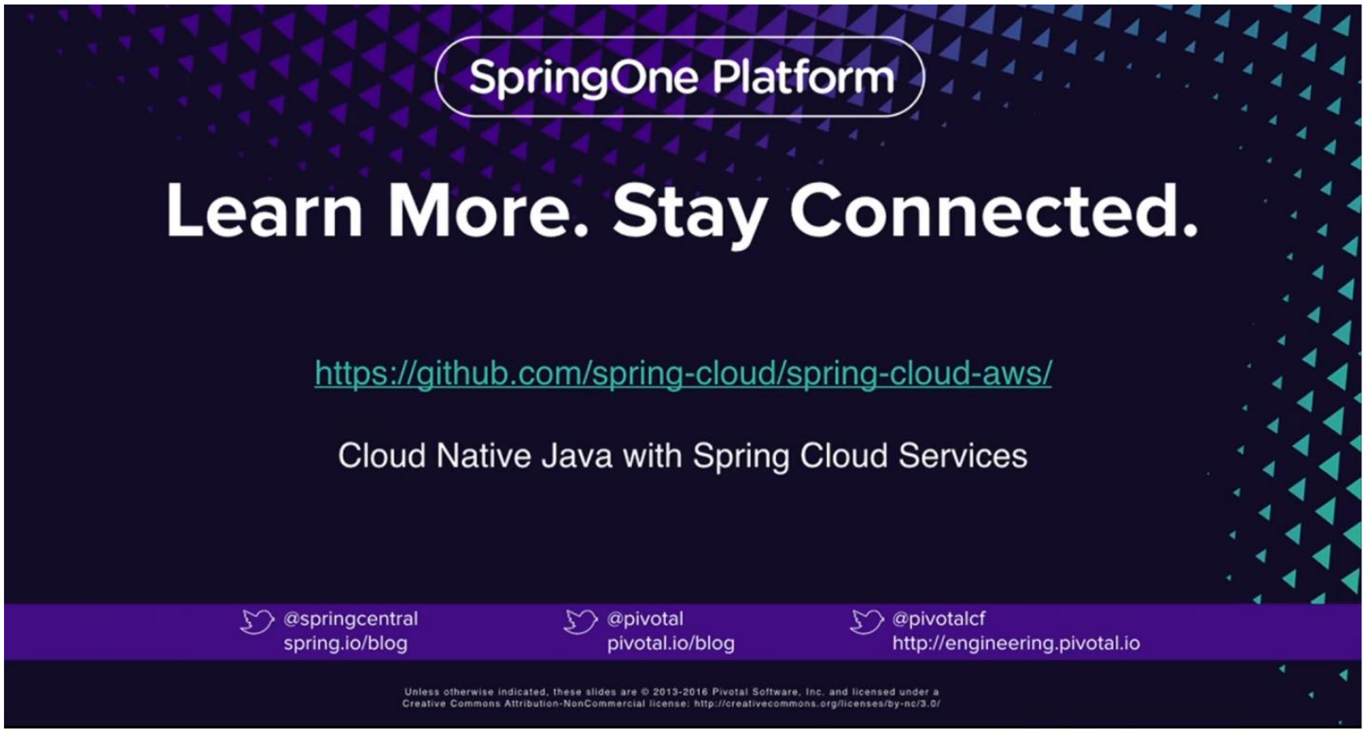
AWS Lambda Support

```java
public class LambdaExample implements RequestHandler<S3EventNotification, String> {

     @Override
     public String handleRequest(S3EventNotification input, Context context) {
          return "result";
     }
}
```

```java
@Service
public class LambdaTest {

     @LambdaFunction
     public String receiveNotification(S3EventNotification input) {
          return "test";
     }
}
```

Lambda Function Structure





SpringOne Platform

Learn More. Stay Connected.

https://github.com/spring-cloud/spring-cloud-aws/

Cloud Native Java with Spring Cloud Services

@springcentral
spring.io/blog

@pivotal
pivotal.io/blog

@pivotalcf
http://engineering.pivotal.io