Announcing Open Source gRPC Kotlin

James Ward, Developer Advocate, Google Cloud
@_JamesWard

What does Kotlin look like?

KOTLIN

```
class Order : Service() {

    override fun onCreate(savedConfiguration: Configuration?) {
        . . .

        worker.setOnCompleteListener { status ->
            Logger.log(Level.INFO, "Completed phase $status")
        }
    }
}
```

Nullable and NonNull built into type system

First class support for lambdas

Use template expressions in strings to avoid concatenation

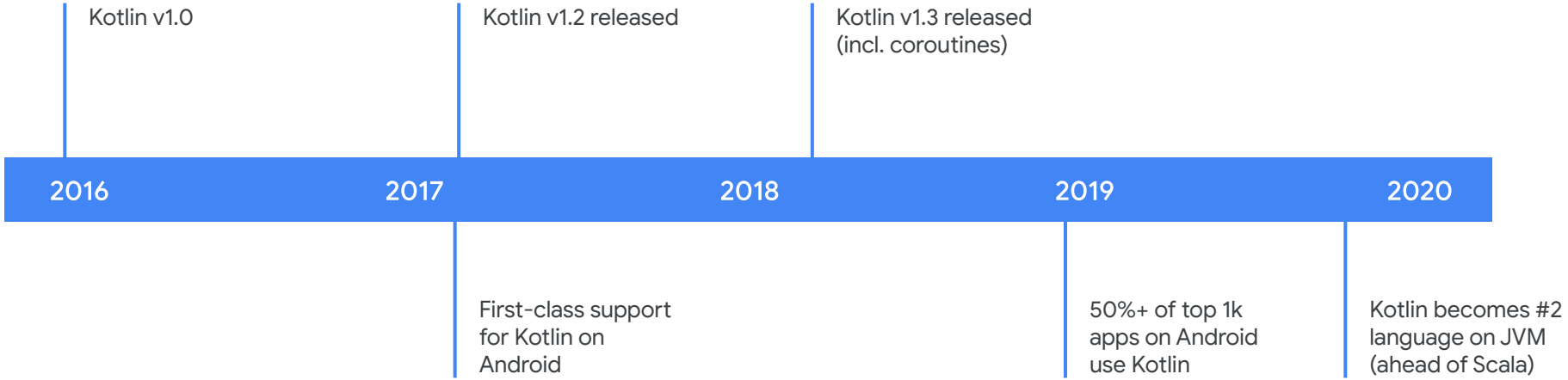
Semicolons are optional

Plus much more...

- Coroutines for background task management
- Bidirectional interoperability with Java
- Expressive / concise code
- Type-safe builders / DSLs

Why Kotlin?

Kotlin took hold on Android but now growing on the server



Data classes: PersonJava

```java
public final class PersonJava {

   private final String name;
   private final int age;
   private final String email;
   private final long phone;

   public PersonJava(String name, int age, String email, long phone) {
       this.name = name;
       this.age = age;
       this.email = email;
       this.phone = phone;
   }

   public String getName() {
       return name;
   }

   public int getAge() {
       return age;
   }

   public String getEmail() {
       return email;
   }

   public long getPhone() {
       return phone;
   }

   @Override
   public String toString() {
       return "PersonJava{" +
               "name='" + name + '\'' +
               ", age=" + age +
               ", email='" + email + '\'' +
               ", phone=" + phone +
               '}';
   }

   @Override
   public boolean equals(Object o) {
       if (this == o) {
           return true;
       }
       if (o == null || getClass() != o.getClass()) {
           return false;
       }

       PersonJava that = (PersonJava) o;

       if (age != that.age) {
           return false;
       }
       if (phone != that.phone) {
           return false;
       }
       if (name != null ? !name.equals(that.name) : that.name != null) {
           return false;
       }
       return email != null ? email.equals(that.email) : that.email == null;
   }

   @Override
   public int hashCode() {
       int result = name != null ? name.hashCode() : 0;
       result = 31 * result + age;
       result = 31 * result + (email != null ? email.hashCode() : 0);
       result = 31 * result + (int) (phone ^ (phone >>> 32));
       return result;
   }
}
```

Data classes: PersonKotlin

```
data class PersonKotlin(val name: String, val age: Int, val email: String, val phone: Long)
```

Protos + BiDi Streaming gRPC

- Language Agnostic / Mix&Match
- Protocol Evolution
- Strongly-typed
- Generate serializers
- Generate service stubs
- Generate client stubs
- Fast Network RPC!

```
// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

gRPC Remote Procedure Call framework

HTTP/2

Protobuffer 3

HTTP/2

Binary Protocol - no more Text

Native Stream Support - no need for WebSocket

Stream Multiplexing - single connection

Header compression - saves more bandwidth

Kotlin + gRPC

https://github.com/grpc/grpc-kotlin

gRPC Kotlin

- Generates Kotlin-friendly gRPC stubs
- Built on gRPC Java
- Coroutines for async
- Kotlin Flow API for streams
- Plugs into Gradle & Maven builds via existing protobuf plugins
- Available today (0.1.1) in Maven Central

hello, world demo

Gradle Setup

Protobuf Plugin: com.google.protobuf:protobuf-gradle-plugin:0.8.12

Stub Dependency: io.grpc:grpc-kotlin-stub:0.1.1

Protobuf Plugin Config:

```
protobuf {
    protoc { artifact = "com.google.protobuf:protoc:${protobuf_version}" }
    plugins {
        grpc { artifact = "io.grpc:protoc-gen-grpc-java:1.28.1" }
        grpckt { artifact = "io.grpc:protoc-gen-grpc-kotlin:0.1.1" }
    }
    generateProtoTasks {
        all().each { task ->
            task.plugins {
                grpc { }
                grpckt { }
            }
        }
    }
}
```

Hello, World

```
// Server
private class HelloWorldService: GreeterCoroutineImplBase() {
 override suspend fun sayHello(request: HelloRequest) = HelloReply
     .newBuilder()
     .setMessage("Hello ${request.name}")
     .build()
}

// Client
val request = HelloRequest.newBuilder().setName(name).build()
val response = async { stub.sayHello(request) }
println("Received: ${response.await().message}")
```

Kotlin Protobufs (coming soon...)

```
// Server
private class HelloWorldService: GreeterCoroutineImplBase() {
 override suspend fun sayHello(request: HelloRequest) = helloReply {
     message = "Hello ${request.name}"
 }
}

// Client
val request = helloRequest { name = "World!" }
val response = async { stub.sayHello(request) }
println("Received: ${response.await().message}")
```

Maven Plugin Setup

```
<groupId>org.xolstice.maven.plugins</groupId>
<artifactId>protobuf-maven-plugin</artifactId>
<version>0.5.0</version>
<configuration>
  <protocArtifact>com.google.protobuf:protoc:${protobuf.version}:exe:${os.detected.classifier}</protocArtifact>
</configuration>
<executions>
  <execution>
    <id>grpc-kotlin</id>
    <goals>
      <goal>compile</goal>
      <goal>compile-custom</goal>
    </goals>
    <configuration>
      <pluginId>grpc-kotlin</pluginId>
      <pluginArtifact>io.grpc:protoc-gen-grpc-kotlin:0.1.1:exe:${os.detected.classifier}</pluginArtifact>
    </configuration>
  </execution>
</executions>
```

Server Streaming

```
// Proto
service Greeter {
  rpc SayHelloStream (HelloRequest) returns (stream HelloReply) {}
}

// Server
private class HelloWorldService:
  GreeterGrpcKt.GreeterCoroutineImplBase() {
  override fun sayHelloStream(req: HelloRequest): Flow<HelloReply> = flow {
    while (true) {
      delay(1000)
      emit(HelloReply.newBuilder().setMessage("Hello, ${req.name}").build())
    }
  }
}

// Client
val req = HelloRequest.newBuilder().setName(name).build()
stub.sayHelloStream(req).collect { response ->
  println(response.message)
}
```

BiDi Streaming

```
// Server
private class HelloWorldService : GreeterGrpcKt.GreeterCoroutineImplBase() {
  override fun sayHelloStream(requests: Flow<HelloRequest>): Flow<HelloReply> {
    return requests.map { request ->
      HelloReply.newBuilder().setMessage("hello, ${request.name}").build()
    }
  }
}

// Client
val helloFlow = flow { while(true) {
    delay(1000)
    emit(HelloRequest.newBuilder().setName("world").build())
}}

stub.sayHelloStream(helloFlow).collect { helloResponse ->
  println(helloResponse.message)
}
```

grpc.io/docs/quickstart/kotlin/

github.com/GoogleCloudPlatform/kotlin-samples/tree/master/run