Wojciech Mazur
Scala Tooling Engineer

Scala 3 Compiler Team &
maintainer of Scala Native

VIRTUS LAB

Scala Native in the world of serverless

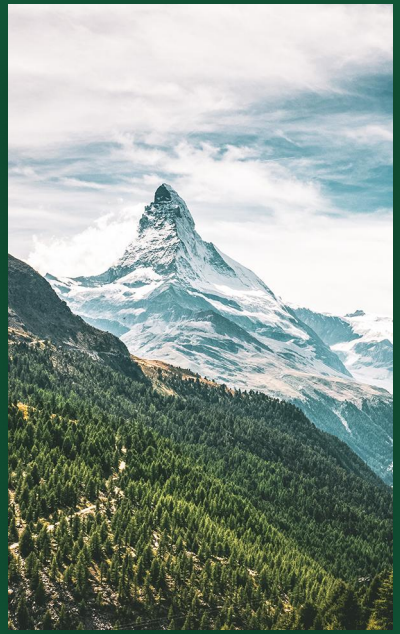
... or how to make it work?

In the AWS... 

...without dedicated runtime

...and without Java SDK

VIRTUSLAB



01 **No runtime?**

Hold my lambda (handler)!

02 No SDK?

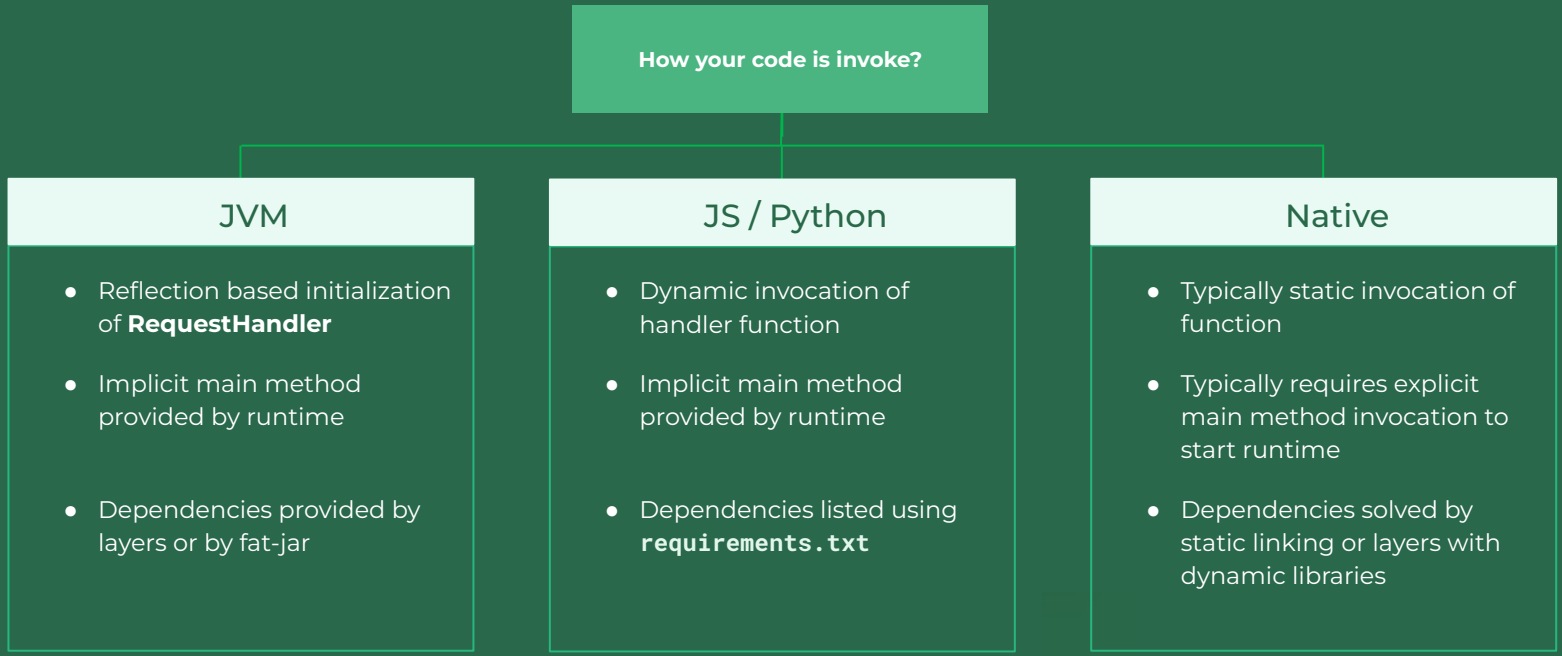
I'll build my own SDK, with cats, and effects

03 So what's next?

VIRTUSLAB

AWS Lambda runtime

How it's executing?



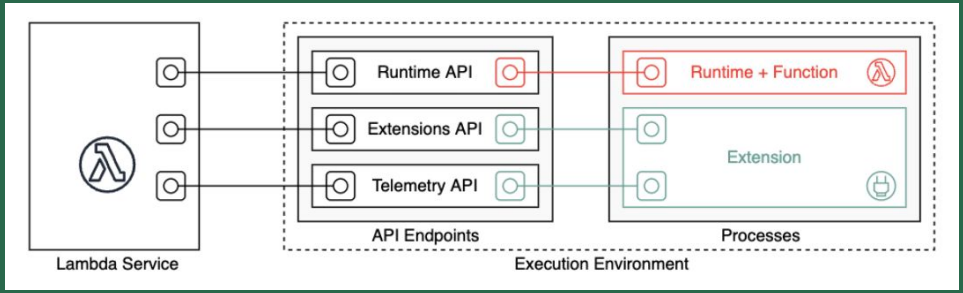
Runtimes based on Amazon Linux containers

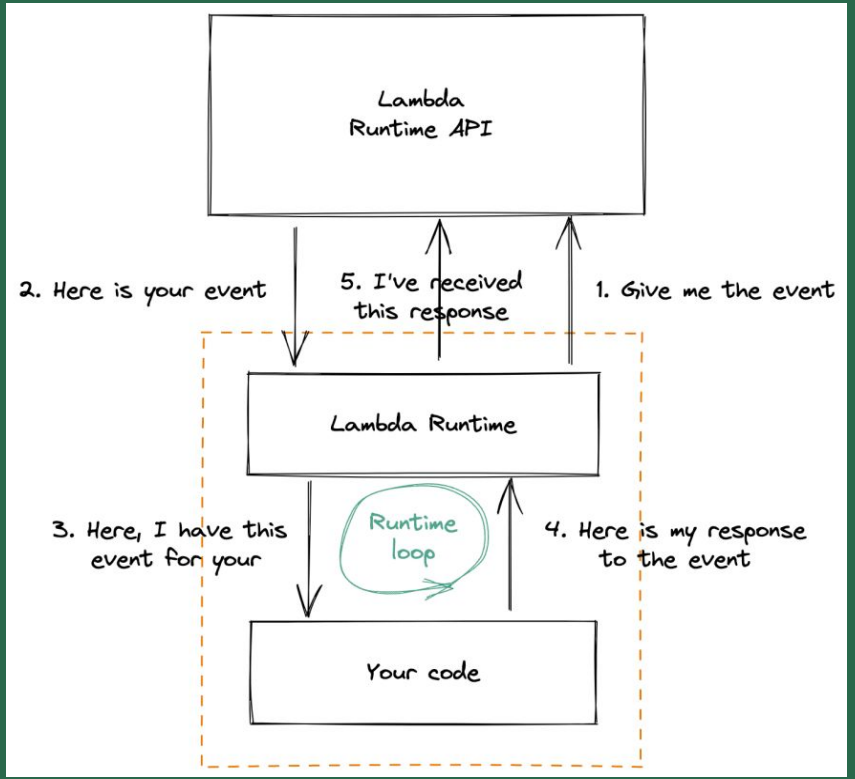
The ultimate entrypoint: /var/task/bootstrap

VIRTUSLAB

AWS Lambda runtime

How it works?





VIRTUSLAB

AWS Lambda runtime

Providing a custom runtime

```scala
def lambdaHandler[I: Reader, O: Writer](event: I)(using Context): O = ???

trait AWSRuntime[I: Reader, O: Writer]:
  def start() = while(true){
    (for
       nextInput     <- http.get(s"http://$AWS_LAMBDA_RUNTIME_API/runtime/invocation/next")
       event         <- read[I](nextInput.body)
       given Context <- readContext(nextInput.headers)
       response      <- lambdaHandler(event)
     do
       http.post(s"http://$AWS_LAMBDA_RUNTIME_API/runtime/invocation/${ctx.requestId}/response")(
         body = write(response)
       )
    ).recover: ex =>
       val error = failureContext(ex)
       http.post(s"http://$AWS_LAMBDA_RUNTIME_API/runtime/invocation/${ctx.requestId}/error")(
         body = write(errorResponse)
       )
```

Pseudo-code for Scala AWS Runtime

AWS Lambda runtime

Providing a custom runtime

```scala
trait LambdaHandler[I: Reader, O: Writer]:
  def run(event: I)(using Context): O
  private val _ = LambdaHandler.run(this)

object LambdaHandler:
  inline def apply[I: Reader, O: Writer](inline handler: I => Context ?=> O) =
    new LambdaHandler[I, O]:
      override def run(event: I)(using Context): O = handler(event)

  def run[I: Reader, O: Writer](handler: LambdaHandler[I, O]) = ???
```

```scala
import runtime.*
import upickle.default.*

case class Event(text: String) derives Reader
case class Result(status: String) derives Writer

@main def HelloWorld = LambdaHandler:
  (event: Event) =>
    val msg = s"GOT REQUEST ${context.getAwsRequestId()} with event data: ${event.text}"
    Result(msg)

```

AWS Lambda layers

Code does not live in the void...

37 runtime dynamic dependencies for hello world

Amazon Linux could contain outdated GCC C stdlib

```
native-lamda git:(master) ✗ ldd .aws-sam/build/HelloWorld/lambdaHandler
        linux-vdso.so.1 (0x00007ffed3f9c000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f74699e4000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f74699df000)
        libidn2.so.0 => /lib/x86_64-linux-gnu/libidn2.so.0 (0x00007f74699be000)
        libcurl.so.4 => /lib/x86_64-linux-gnu/libcurl.so.4 (0x00007f7469917000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f7469600000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f746982e000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f74695e0000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f7469200000)
        libunistring.so.2 => /lib/x86_64-linux-gnu/libunistring.so.2 (0x00007f7469436000)
        libnghttp2.so.14 => /lib/x86_64-linux-gnu/libnghttp2.so.14 (0x00007f74691d6000)
        librtmp.so.1 => /lib/x86_64-linux-gnu/librtmp.so.1 (0x00007f74691b7000)
        libssh.so.4 => /lib/x86_64-linux-gnu/libssh.so.4 (0x00007f746914a000)
        libpsl.so.5 => /lib/x86_64-linux-gnu/libpsl.so.5 (0x00007f7469136000)
        libssl.so.3 => /lib/x86_64-linux-gnu/libssl.so.3 (0x00007f7469092000)
        libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x00007f7468c00000)
        libgssapi_krb5.so.2 => /lib/x86_64-linux-gnu/libgssapi_krb5.so.2 (0x00007f7468bac000)
        libldap-2.5.so.0 => /lib/x86_64-linux-gnu/libldap-2.5.so.0 (0x00007f7468b4d000)
        liblber-2.5.so.0 => /lib/x86_64-linux-gnu/liblber-2.5.so.0 (0x00007f7469081000)
        libzstd.so.1 => /lib/x86_64-linux-gnu/libzstd.so.1 (0x00007f7468a7e000)
        libbrotlidec.so.1 => /lib/x86_64-linux-gnu/libbrotlidec.so.1 (0x00007f7469428000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f7469065000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f7469a02000)
        libgnutls.so.30 => /lib/x86_64-linux-gnu/libgnutls.so.30 (0x00007f7468893000)
        libhogweed.so.6 => /lib/x86_64-linux-gnu/libhogweed.so.6 (0x00007f746884b000)
        libnettle.so.8 => /lib/x86_64-linux-gnu/libnettle.so.8 (0x00007f7468805000)
        libgmp.so.10 => /lib/x86_64-linux-gnu/libgmp.so.10 (0x00007f7468783000)
        libkrb5.so.3 => /lib/x86_64-linux-gnu/libkrb5.so.3 (0x00007f74686b8000)
        libk5crypto.so.3 => /lib/x86_64-linux-gnu/libk5crypto.so.3 (0x00007f7468689000)
        libcom_err.so.2 => /lib/x86_64-linux-gnu/libcom_err.so.2 (0x00007f746905b000)
        libkrb5support.so.0 => /lib/x86_64-linux-gnu/libkrb5support.so.0 (0x00007f746904d000)
        libsasl2.so.2 => /lib/x86_64-linux-gnu/libsasl2.so.2 (0x00007f746866e000)
        libbrotlicommon.so.1 => /lib/x86_64-linux-gnu/libbrotlicommon.so.1 (0x00007f746864b000)
        libp11-kit.so.0 => /lib/x86_64-linux-gnu/libp11-kit.so.0 (0x00007f7468510000)
        libtasn1.so.6 => /lib/x86_64-linux-gnu/libtasn1.so.6 (0x00007f74684f8000)
        libkeyutils.so.1 => /lib/x86_64-linux-gnu/libkeyutils.so.1 (0x00007f7469044000)
        libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x00007f74684e4000)
        libffi.so.8 => /lib/x86_64-linux-gnu/libffi.so.8 (0x00007f74684d7000)
```

```
+ /var/task/HelloWorld
/var/task/HelloWorld: /lib64/libm.so.6: version `GLIBC_2.29' not found (required by /var/task/HelloWorld)
/var/task/HelloWorld: /opt/sn-runtime/lib/libcurl.so.4: no version information available (required by /var/task/HelloWorld)
/var/task/HelloWorld: /lib64/libc.so.6: version `GLIBC_2.32' not found (required by /var/task/HelloWorld)
/var/task/HelloWorld: /lib64/libc.so.6: version `GLIBC_2.34' not found (required by /var/task/HelloWorld)
/var/task/HelloWorld: /lib64/libc.so.6: version `GLIBC_2.33' not found (required by /var/task/HelloWorld)
END RequestId: 0ce3ce84-41bf-449b-bf75-5702b78d3d20
```

AWS Lambda runtime

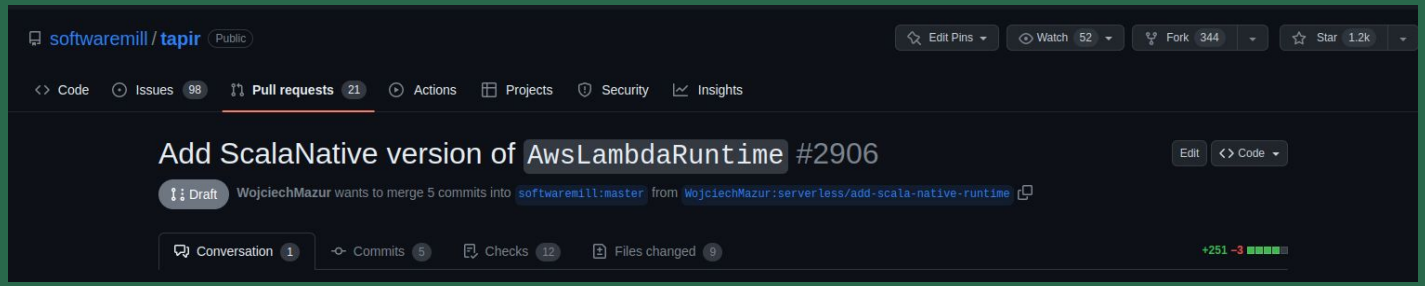
Tapir based runtime

```scala
import cats.effect.*
import sttp.tapir.*
import sttp.tapir.serverless.aws.lambda.*
import sttp.tapir.serverless.aws.lambda.runtime.*
import sttp.tapir.generic.auto.*
import sttp.tapir.json.circe.*

case class Response(msg: String)

object HttpExample extends AwsLambdaIORuntime {
  val helloEndpoint: ServerEndpoint[Any, IO] = endpoint.get
    .in("api" / "hello" / paths)
    .errorOut(stringBody)
    .out(jsonBody[Response])
    .serverLogic { args =>
      val name = args.headOption
      IO.pure:
        Response:
          s"Hello ${name.getOrElse("anonymous")}. Welcome to Serverless Lambda!"
        .asRight[String]
    }

  val wildcardEndpoint: ServerEndpoint[Any, IO] = endpoint.get.in(paths).out(stringBody).serverLogic:
    input =>
      val msg = s"Unknown endpoint: ${input.mkString("/")}"
      IO.pure(msg.asRight[Unit])

  override val endpoints = Seq(helloEndpoint, wildcardEndpoint)
  override val serverOptions: AwsServerOptions[IO] = AwsCatsEffectServerOptions.noEncoding[IO]
}

```


VIRTUSLAB

AWS Lambda runtime

WIP: Tapir based native-runtime



VIRTUSLAB

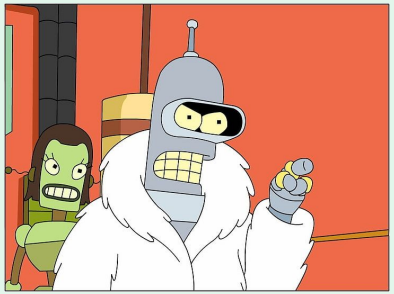
01 No runtime?

Hold my lambda (handler)!

02 No SDK?

I'll build my own SDK, with cats, and effects

03 So what's next?



VIRTUSLAB

Introduction to Smithy4s

What is smithy?

```
namespace smithy4s.hello

use alloy#simpleRestJson

@simpleRestJson
service HelloWorldService {
  version: "1.0.0",
  operations: [Hello]
}

@http(method: "POST", uri: "/{name}", code: 200)
operation Hello {
  input: Person,
  output: Greeting
}

structure Person {
  @httpLabel
  @required
  name: String,

  @httpQuery("town")
  town: String
}

structure Greeting {
  @required
  message: String
}
```

```
package smithy4s.hello

import smithy4s.*

case class Person(name: String, town: Option[String] = None)
case class Greeting(message: String)

```

```
package smithy4s.hello

import smithy4s.*

trait HelloWorldServiceGen[F[_, _, _, _, _]]: self =>
  def hello(name: String, town: Option[String] = None): F[Person, Nothing, Greeting, Nothing, Nothing]

object HelloWorldServiceGen:
  def apply[F[_]](implicit F: Impl[F]): F.type = F

```

```
package smithy4s

package object hello {
  type HelloWorldService[F[_]] = smithy4s.kinds.FunctorAlgebra[HelloWorldServiceGen, F]
  val HelloWorldService = HelloWorldServiceGen
}
```


VIRTUSLAB

Introduction to Smithy4s

Using generated API?

```scala
import runtime.* // Platform specific IORuntime provider
import org.http4s.ember.client.EmberClientBuilder
import org.http4s.*
import cats.effect.*

import smithy4s.http4s.SimpleRestJsonBuilder
import smithy4s.hello.* // generated

object SmithyExampleClient extends AppRuntime:
  val helloWorldClient: Resource[IO, HelloWorldService[IO]] =
  for
    client ← EmberClientBuilder.default[IO].build
    helloClient ← SimpleRestJsonBuilder(HelloWorldService)
      .client(client)
      .uri(Uri.unsafeFromString("http://localhost:9000"))
      .resource
   yield helloClient

  val run = helloWorldClient.use: client ⇒
    client.hello("Sam", Some("New York City"))
      .flatMap(greeting ⇒ IO.println(greeting.message))

```

```scala
import runtime.* // Platform specific IORuntime provider
import smithy4s.hello.* // Generated
import smithy4s.http4s.SimpleRestJsonBuilder

import cats.effect.*
import org.http4s.*
import org.http4s.ember.server.*

object HelloWorldImpl extends HelloWorldService[IO]:
  def hello(name: String, town: Option[String]): IO[Greeting] = IO.pure:
    town match
      case None ⇒ Greeting(s"Hello $name!")
      case Some(t) ⇒ Greeting(s"Hello $name from $t!")

object Routes:
  val example: Resource[IO, HttpRoutes[IO]] =
    SimpleRestJsonBuilder
      .routes(HelloWorldImpl)
      .resource

object SmithyExampleServer extends AppRuntime:
  val run = Routes.example
    .flatMap { routes ⇒
      EmberServerBuilder
        .default[IO]
        .withPort(port"9000")
        .withHost(host"localhost")
        .withHttpApp(routes.orNotFound)
        .build
    }
    .use(_ ⇒ IO.never)

```

Providing our own SDK

Generating and using AWS API clients

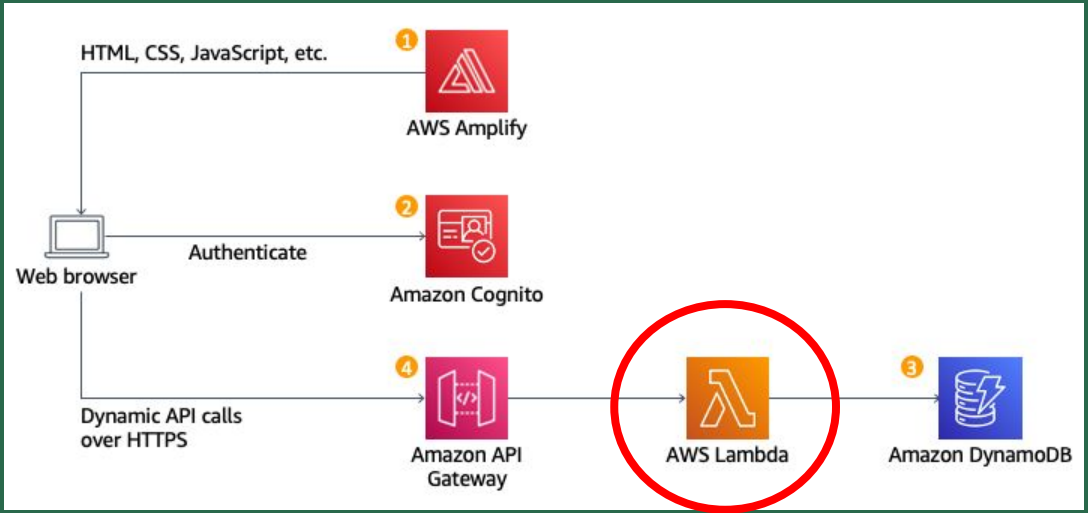
```
cs install smithy4s \
  --channel https://disneystreaming.github.io/coursier.json

smithy4s generate \
  --dependencies com.disneystreaming.smithy:aws-dynamodb-spec:2023.02.10 \
  -o outputDir
```

```scala
import com.amazonaws.dynamodb.* // Generated by Smithy4s
import smithy4s.aws.*
import smithy4s.aws.http4s.AwsHttp4sBackend

// Defined in Java SDK or reimplemented in pure Scala for Native
import com.amazonaws.services.lambda.runtime.Context

// Important:
// Default CatsEffect 3.5 IORuntimefor native can deadlock
// when used with AwsHttp4sBacnend + Ember.
// Use custom EpoolRuntime instead
//> using dep com.armanbilge::epollcat::0.1.4
given IORuntime = EpollRuntime.global

val dynamoDbClient = for
  httpClient ← EmberClientBuilder.default[IO].build
  dynamodb ← DynamoDB.simpleAwsClient(httpClient, AwsRegion.EU_CENTRAL_1)
 yield dynamodb

def recordRide(rideId: String, username: String): IO[Unit] =
  dynamoDbClient.use:
     _.putItem(
        tableName = TableName("Rides"),
        item = Map(
          "rideId" → rideId,
          "user" → username,
          "requestTime" → LocalDateTime.now().toString()
        ).map: (k, v) ⇒
          AttributeName(k) → AttributeValue.SCase(StringAttributeValue(v))
      ).flatMap(res ⇒ IO.println(s"Recorded ride with id=$rideId, res=$res"))

```

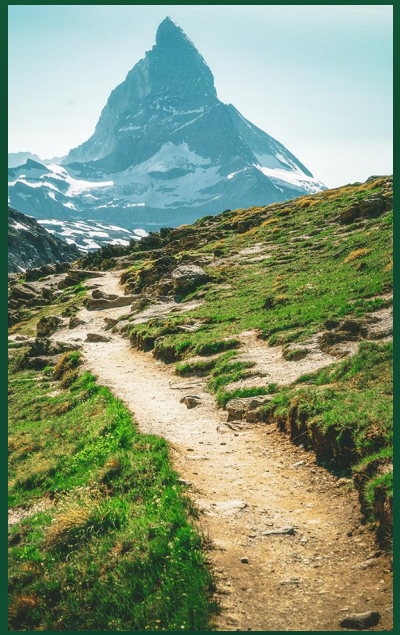

VIRTUSLAB

Business problem to solve

Unicorn delivery app



Based on AWS workshop:
https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/



01 No runtime?

Hold my lambda (handler)!

02 No SDK?

I'll build my own SDK, with cats, and effects

03 So what's next?

VIRTUSLAB

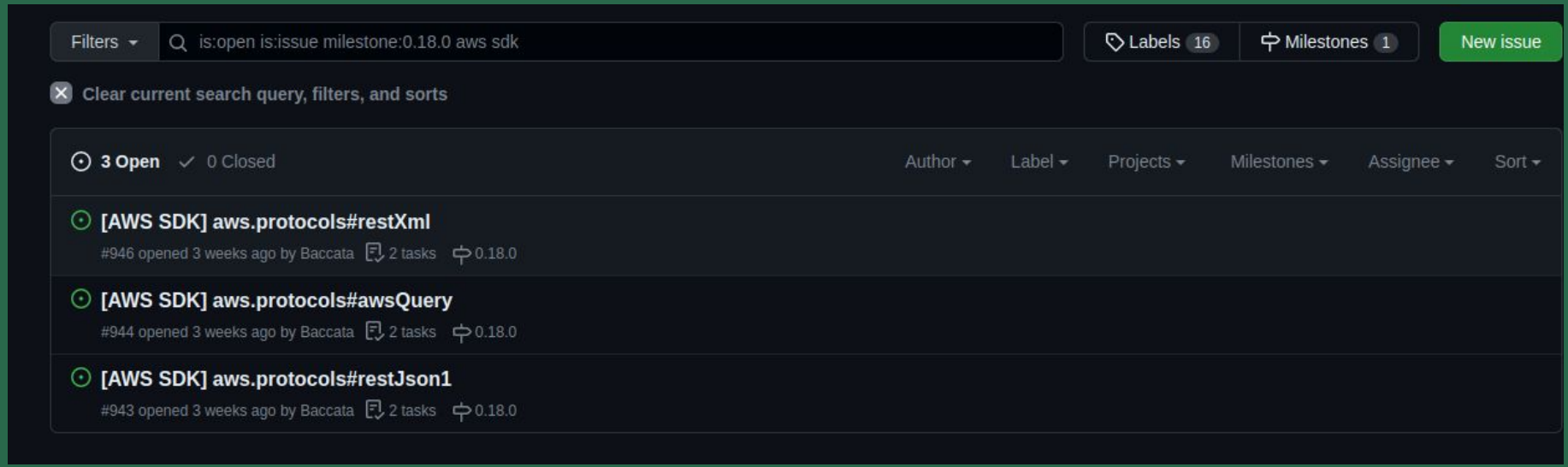
Current state

Experimental

“I love the POC, but let’s be real: this is a toy and not something a Scala shop can deploy to production.”

Next steps for serverless

Better AWS integration



VIRTUSLAB

Next steps for serverless

Runtime improvements

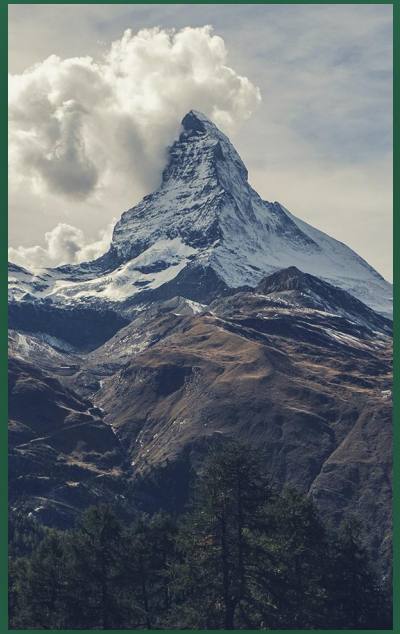
Stable Scala Native runtime for AWS
Possibly reflection based

Tooling support
SAM like tool, or generation of Makefiles/templates

Resolving vendor locking hazard
CodeGen for Google Cloud gRPC services and Azure clients

Scala Native DX
Faster build times using incremental builds, smaller binaries

Private cloud solutions
Knative, OpenFaaS, Fission, etc



Thank you!

Ask me anything!

WojciechM_dev

wmazur@virtuslab.com

VIRTUSLAB