Rust on AWS Lambda

Ryan Scott Brown

Agenda

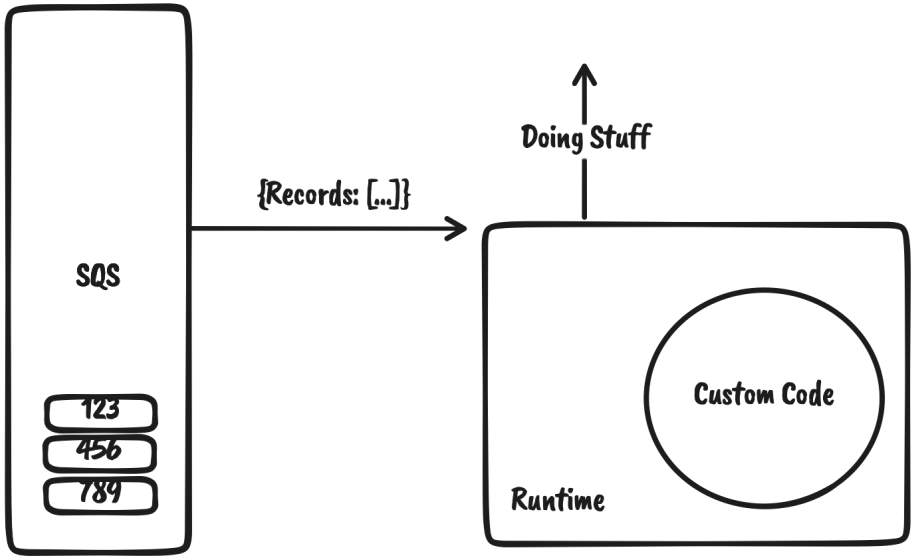
- What’s a Lambda, Will It Rust™
- Building for Lambda
- Deploying to Lambda
- Our first application
- APIs with Smithy
- Storage and queries

What's a Lambda

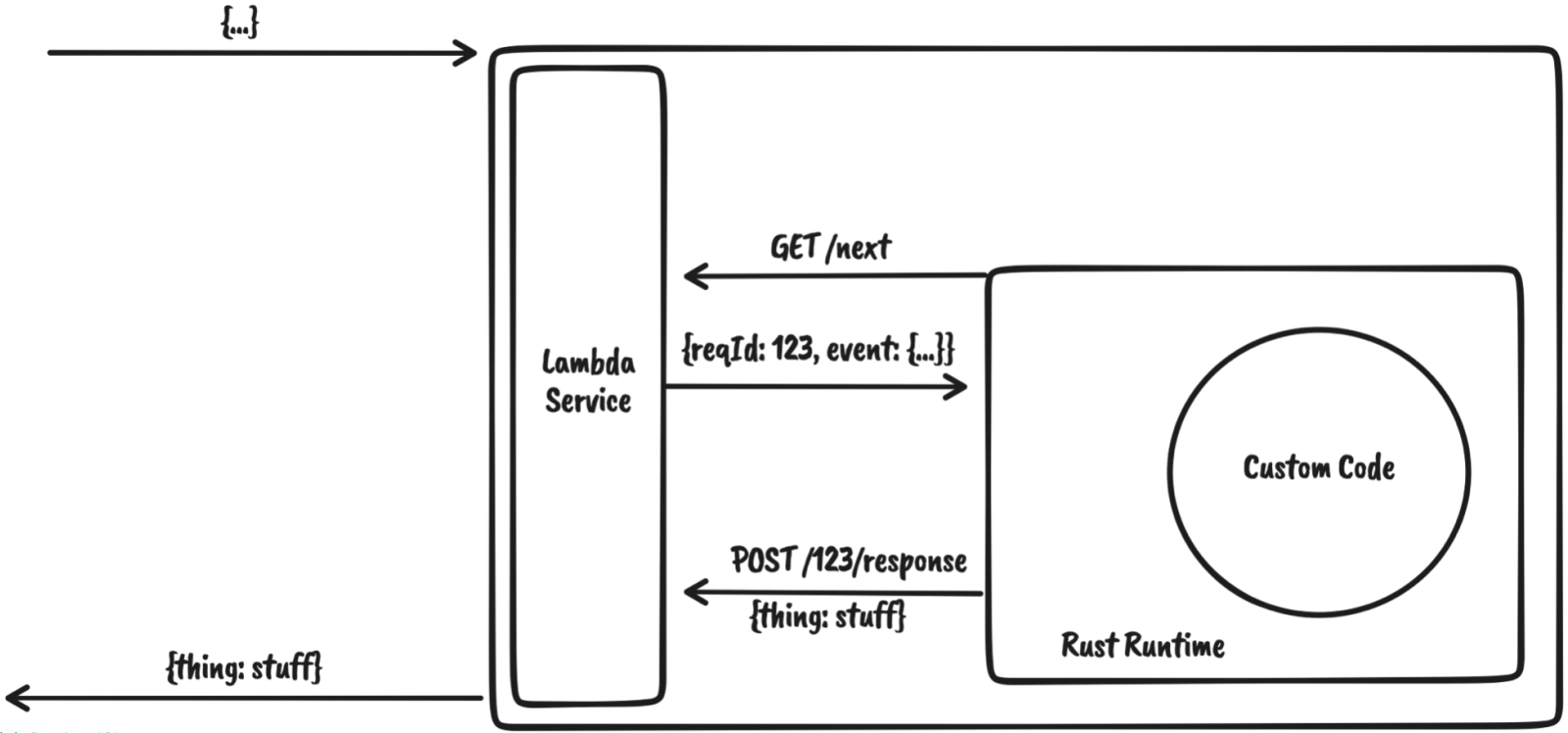
- Runs code on-demand in response to events
  - API Gateway, Kinesis, SQS, S3 object access and write, DynamoDB CDC, Alexa Skills, Amazon Connect activity, Cognito logins, and more
- Trades persistence for reduced management burden
- Native support for NodeJS, Python, Ruby, Java, Go, .NET, and .NET Core

Event Sources


```
{
  "Records": [
    {
      "messageId": "059f36b4-87a3-44ab-83d2-661975830a7d",
      "receiptHandle":
"AQEBwJnKyrHigUMZj6rYigCgxlaS3SLy0a...",
      "body": "Test message.",
      "attributes": {
        "ApproximateReceiveCount": "1",
        "SentTimestamp": "1545082649183",
        "SenderId": "AIDAIENQZJOLO23YVJ4VO",
        "ApproximateFirstReceiveTimestamp": "1545082649185"
      },
      "messageAttributes": {},
      "md5OfBody": "e4e68fb7bd0e697a0ae8f1bb342846b3",
      "eventSource": "aws:sqs",
      "eventSourceARN":
"arn:aws:sqs:us-east-2:123456789012:my-queue",
      "awsRegion": "us-east-2"
    }
  ]
}
```

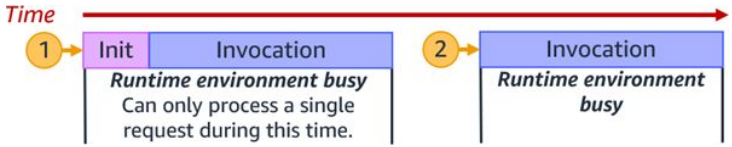



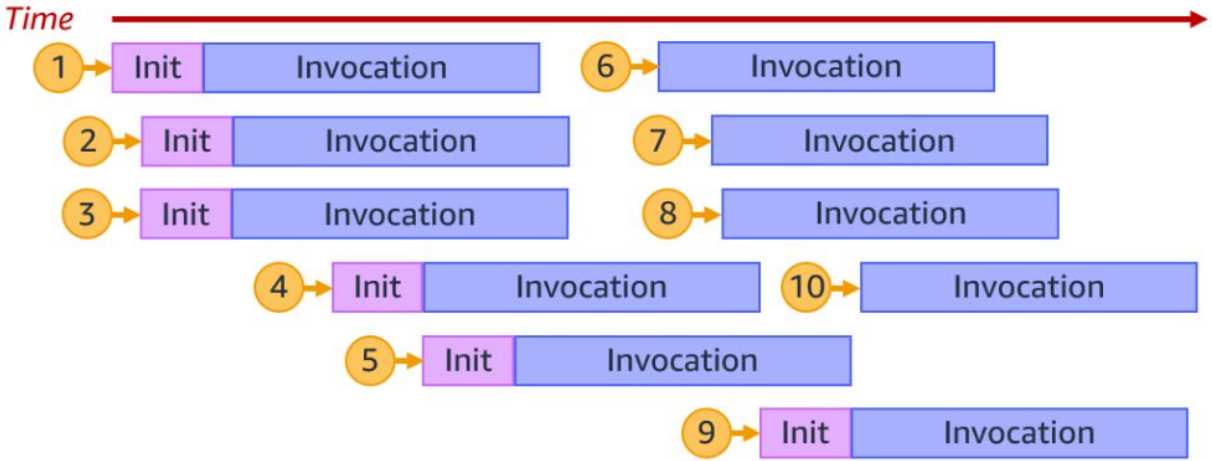
Using a Runtime



Docs: Lambda Runtime API

Containers and Cold-Starts





Blog: Understanding Lambda Concurrency

HTTP `GET /` Endpoint

Generating SerDe Types

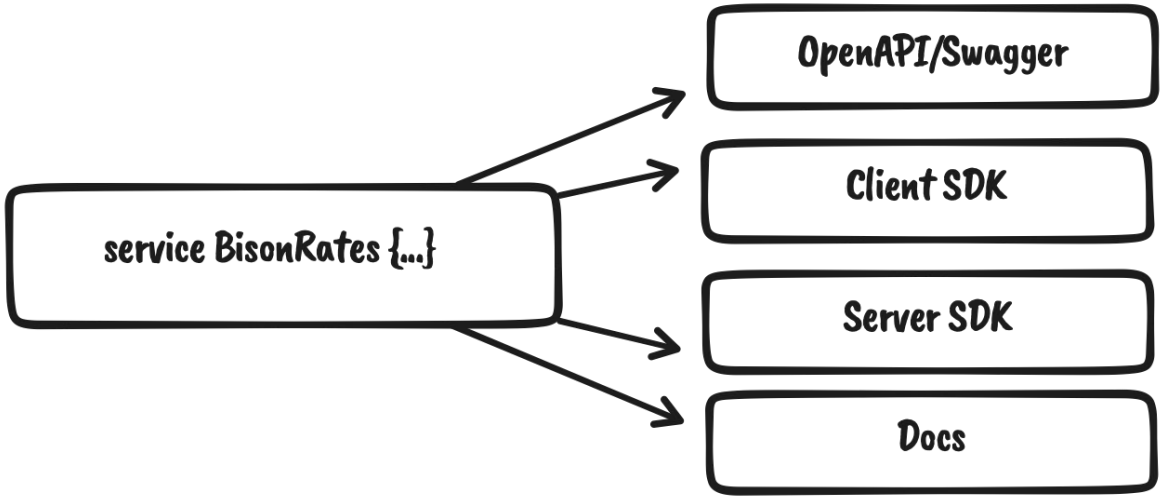
```
#[serde(Deserialize, Serialize)]
struct NewBisonRequest {
    name: String
    herd: String
}
```

Boilerplate is boring, and doesn’t add anything… what if we could do it once and generate the code?

Smithy

```
// model/get.smithy
namespace gov.ny.buffalo

@http(uri: "/{stage}/bison/{herd}", method: "GET")
@readonly
operation ListBison {
    input: ListBisonInput,
    output: ListBisonOutput
}

@input
structure ListBisonInput {
    @required
    @httpLabel
    herd: Name,

    @required
    @httpLabel
    stage: String,
}
```

Smithy SDL



Server SDK Code Generator

- Automatic ...
  - Requests
  - Responses
  - Errors
  - Routes
- Gross types for builders

```
impl<B, Op0, In0, Op1, In1> Default for OperationRegistryBuilder<B, Op0,
In0, Op1, In1> {
    fn default() -> Self {
        Self {
            create_bison: Default::default(),
            list_bison: Default::default(),
            _phantom: std::marker::PhantomData,
        }
    }
}
```

- TODO: String validations like @length aren't supported yet for the Rust Server SDK

Server SDK Example

Using Errors

Enumerated by `sdk_server_bison_rates::error::CreateBisonError`

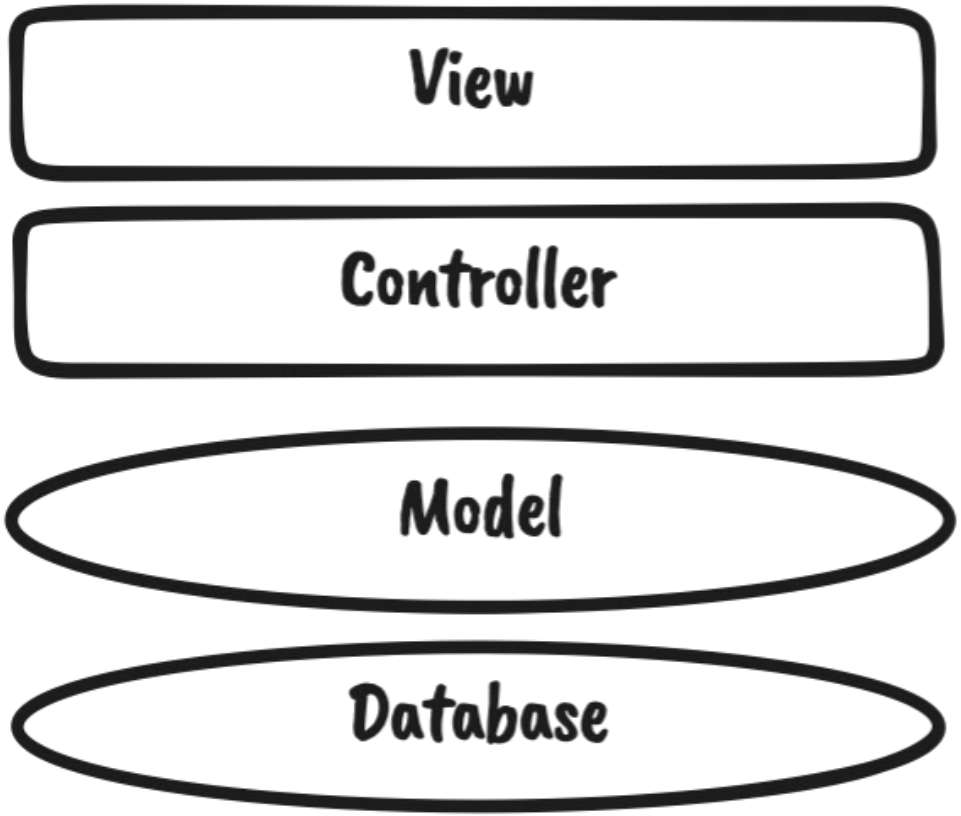
```rust
pub async fn create_bison(data: CreateBisonInput) -> Result<CreateBisonOutput, CreateBisonError> {
    tracing::info!("POST /bison body: {:?}", data);

    if data.name().is_empty() {
        return Err(CreateBisonError::ValidationException(ValidationException {
            message: "Field `name` must have contents".to_string(),
            field_list: None,
        }));
    }
```

Type-enforced error returns, and on the SDK clients the exception types can be auto-generated to avoid stringly-typed error handling.

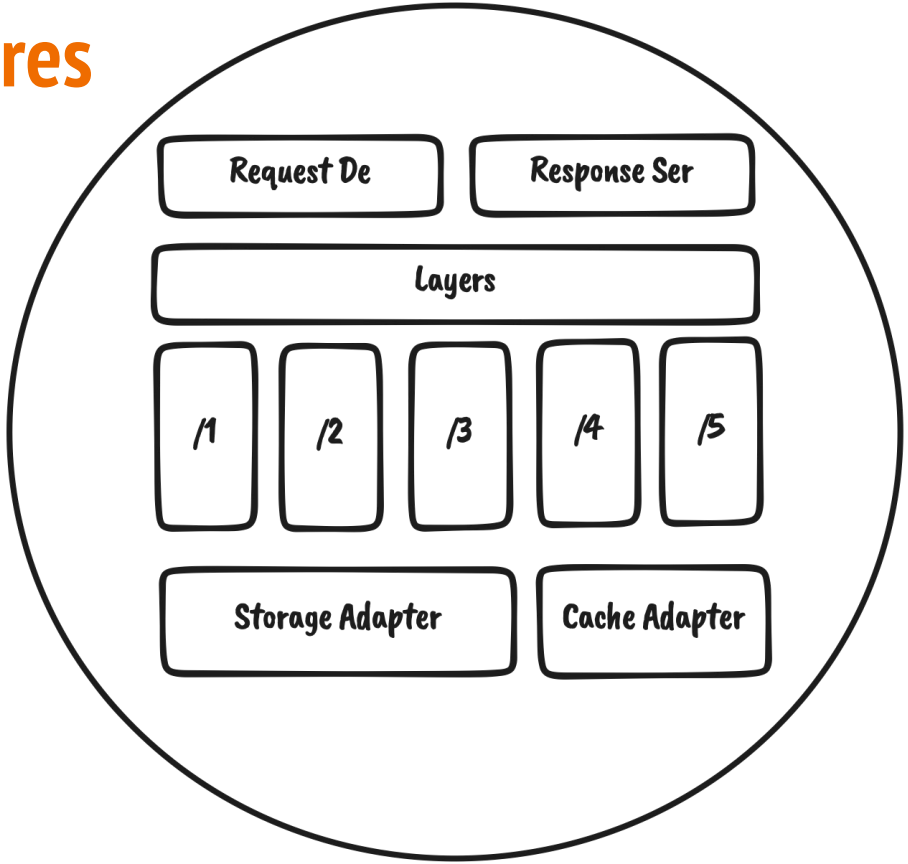
Storing DynamoDB Records

- AWS Rust SDK supports async/await
- Environment credentials
  - See CDK stack for roles
- DynamoDB doesn’t require long-lived connections

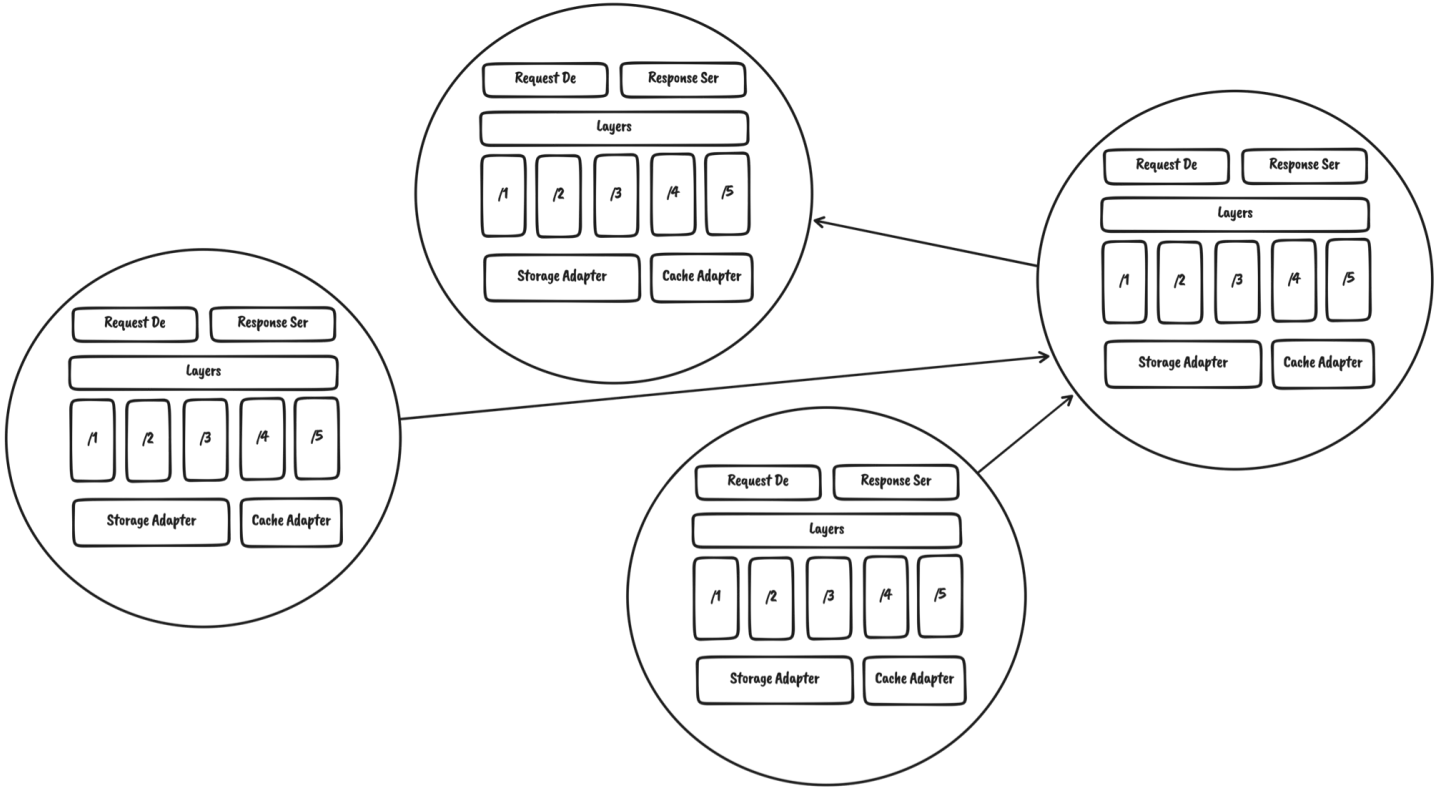
Structuring a Service



Tower Structures



Functions as Segmentation



Fin

Resources

- This code: github.com/ryansb/bison-rates-demo
- AWS Well-Architected Framework Serverless Lens
- Smithy Rust
  - Pull request to fix Server SDK Lambda support (#1338)
- Cloud Development Kit
- Cross-compiling tools
  - cargo-zigbuild
  - cargo-lambda
- The DynamoDB Book
- Boundaries
- Stedi Careers ;)