Kotless

Kotlin Serverless Framework

Vladislav Tankov

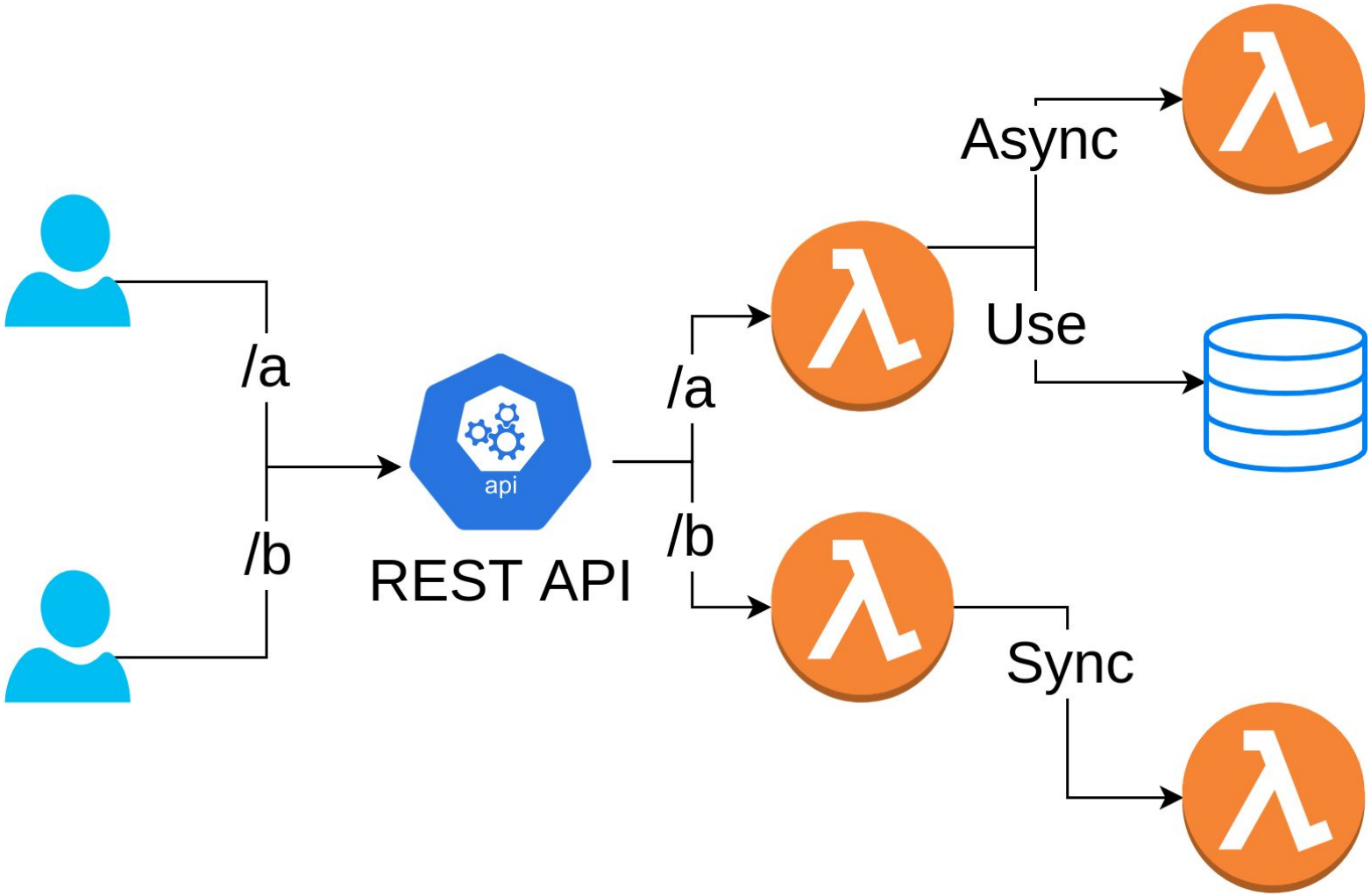
What is serverless?

cloud-computing execution model, in which the cloud provider runs the server and dynamically manages the allocation of machine resources

Wikipedia

How are serverless applications built?

- Three simple steps:
  - Take small elements – stateless functions
  - Compose them with events into an application
  - Deploy it to Cloud runtime


/a
/b
api
REST API
/a
/b
Async
Use
Sync

Still, the idea is pretty simple

Implementation should also be simple, right?

```
@Get("/hello-world")
fun helloWorldRoute(): String {
   return "Hello World"
}
```

People deploy Serverless applications with Infrastructure as Code approach

Infrastructure as Code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools

Wikipedia

With Infrastructure as Code
it is not that simple

Tons of configuration

- 100+ lines for this function
- 1000+ lines for a simple site
- Separate language for configuration

```
resource "aws_lambda_function" "_long" {
 function_name = "Handler__long"
 s3_bucket = "${aws_s3_bucket.ktltst_lambda_s3.bucket}"
 s3_key = "${aws_s3_bucket_object.ktltst_bucket_object.key}"
 source_code_hash = "${base64sha256(file("../build/libs/kotless-dsl-1.0-all.jar"))}"
 handler = "kotless.Lambda::handleRequest"
 runtime = "java8"
 timeout = 30
 role = "${aws_iam_role.ktltst_lambda_role.arn}"
 memory_size = 256
 environment = {
   variables = "${var._long_envvars}"
 }
}

resource "aws_lambda_permission" "_long" {
 statement_id = "AllowAPIGatewayInvoke"
 action = "lambda:InvokeFunction"
 function_name = "${aws_lambda_function._long.arn}"
 principal = "apigateway.amazonaws.com"
 source_arn = "${aws_api_gateway_deployment.ktltst_example_deployment.execution
}

resource "aws_api_gateway_resource" "_long" {
 parent_id = "${aws_api_gateway_rest_api.ktltst_example_rest_api.root_resource_id}"
 rest_api_id = "${aws_api_gateway_rest_api.ktltst_example_rest_api.id}"
 path_part = "long"
}

resource "aws_api_gateway_method" "_long" {
 rest_api_id = "${aws_api_gateway_rest_api.ktltst_example_rest_api.id}"
 resource_id = "${aws_api_gateway_resource._long.id}"
 http_method = "ANY"
 authorization = "NONE"
}

resource "aws_api_gateway_integration" "_long" {
 rest_api_id = "${aws_api_gateway_rest_api.ktltst_example_rest_api.id}"
 resource_id = "${aws_api_gateway_method._long.resource_id}"
 http_method = "${aws_api_gateway_method._long.http_method}"
 depends_on = [
```

Could it be simpler?

```
@Get("/hello-world")
fun helloWorldRoute(): String {
   return "Hello World"
}
```

Infrastructure can be deduced from code

Infrastructure SHOULD be deduced from code

Infrastructure in Code

- Application framework and deployment tool:
  - Write the code with the help of the framework
  - Introspect the code during deployment
  - Create infrastructure and deploy the application
  - Weave the application into the infrastructure in runtime

That is what Kotless does

Actual Kotless code

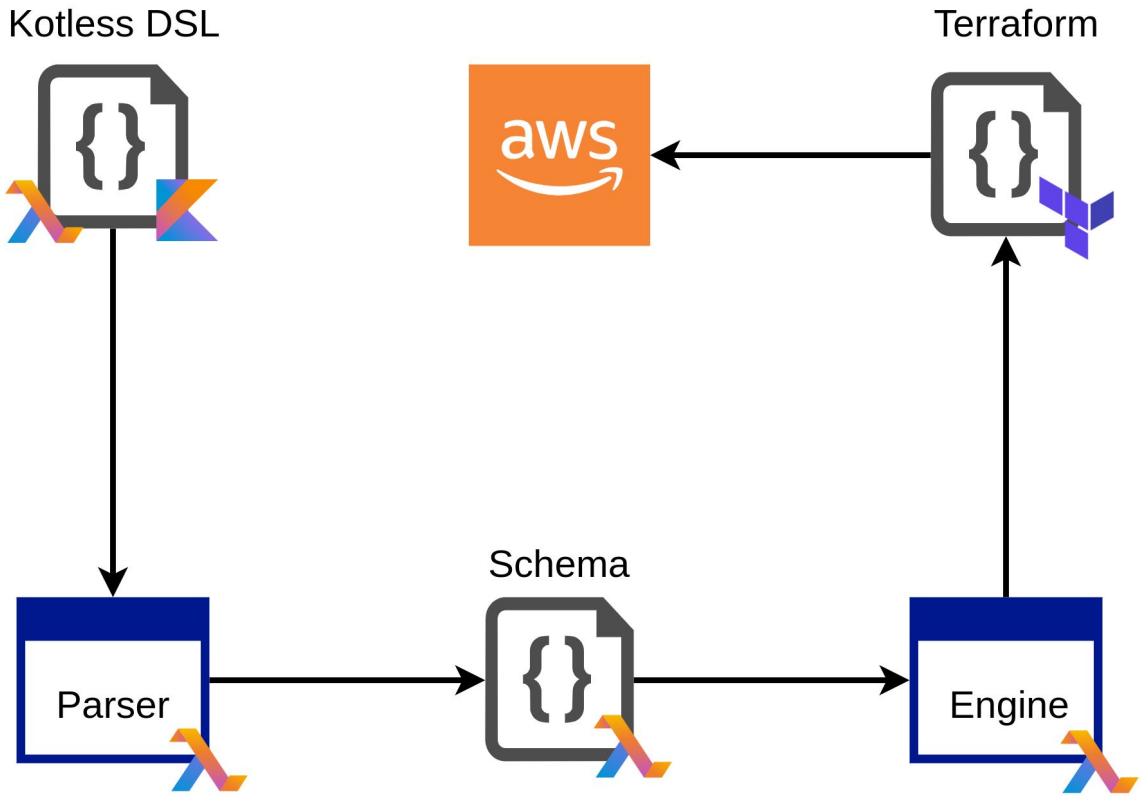
```
@Get("/hello-world")
fun helloWorldRoute(): String {
   return "Hello World"
}
```

Kotless

- Infrastructure in Code tool for Kotlin:
  - Kotless DSL for HTTP events
  - Gradle plugin for deployment
  - Uses Terraform under the hood
  - Supports AWS

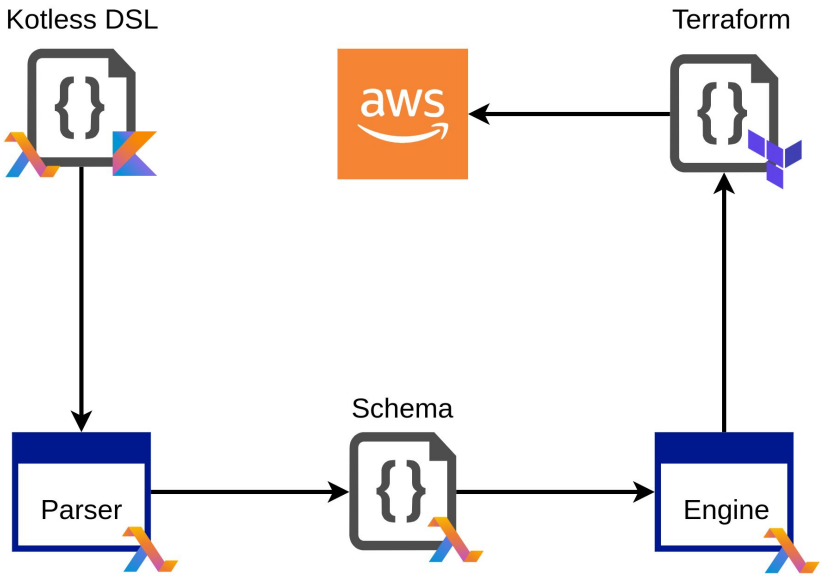
Kotless-based application

How does it work?



What it gives us

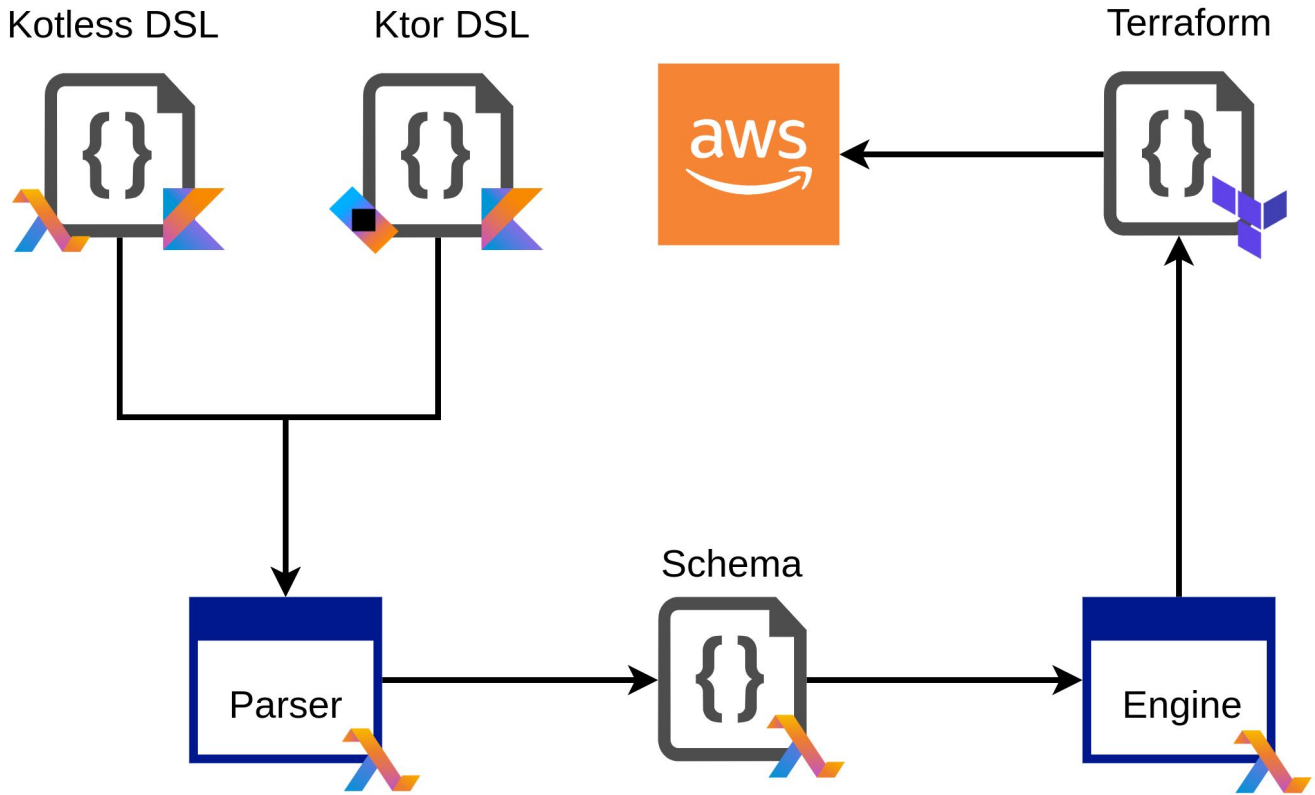
- Cloud agnostic scheme
- Abstraction of deployment
- Abstraction of DSL



Why not support existing framework?

Ktor-based application

Several DSLs



Seamless serverless

- Write the code with any web framework you like
  - Ktor
- Run it locally
- Deploy it
  - As a serverless app to cloud
  - As a standalone app in-house

Let’s go even further

Scheduling?

```
@Scheduled(every5Minutes)
fun scheduledRoute() {
    println("What a lovely day!")
}
```

Permissions?

```
@DynamoDBTable("table", ReadWrite)
object URLStorage {
  fun getByCode(code: String): String? {
      ...
  }
}
```

Code may fully define

- API interaction: *@Get, @Post, ...*
- Events handling: *@Scheduled, ...*
- Permissions requirements: @DynamoDBTable, ...
- *Shared structures: Queue, List, ...*
- *Calls of functions: async { ... }, ...*

Kotless advanced features

What’s next?

More is coming

- We are working hard on:
  - Supporting other clouds
  - Kotless MPP
  - Extended event handling
  - Much more

Give Kotless a try!

github.com/JetBrains/Kotless

vladislav.tankov@jetbrains.com

Remember to vote!

Serverless IS

- Very popular topic:
  - Top-growth cloud service 2nd year
  - Almost as popular as Kotlin on StackOverflow
  - 10 percent less interesting than ML (39% / 49%)
  - Even standardization is on the way
    - CloudEvents.io
    - Knative

Serverless IS

- Applications building approach:
  - Decouple the app as much as possible
  - Make use of cloud provider’s managed services
  - Connect it with the outer world via provider’s API

Why to serverless?

Make the provider manage your infrastructure

Why to serverless

- Benefits all the way:
  - Automatic scaling
  - Fault tolerance
  - Cost
    - Is it?

Why to serverless

- Hundreds of frameworks:
  - Infrastructure as a Code
    - Serverless.com
    - CDK
    - Terraform
  - *Infrastructure in a Code*
    - AWS Chalice
    - Kotless
    - Osiris

Why to serverless

- Availability:
  - Courses
  - Books
  - Samples
  - Researches

The Great Serverless Myth

Serverless is not a thing
without servers

It’s all about management

- Reducing the pain of managing:
  - Scale
  - Price
  - SLA
  - Complexity

Serverless is overcomplicated