Ballerina

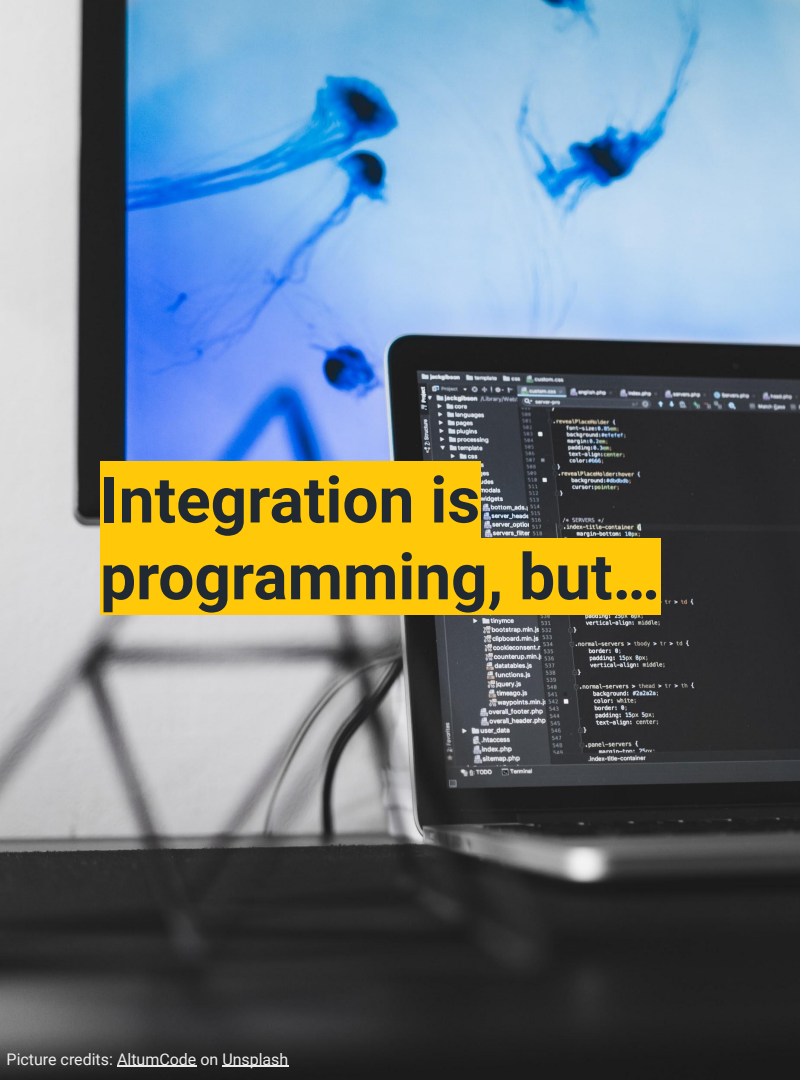
An open-source programming language for the cloud that makes it easier to use, combine, and create network services.

June 2021

The background to Ballerina

15 years and 1000s of customers taught us that …

- All enterprises need lots of integration to innovate digitally
- Integration remains hard, time consuming and expensive
  - This is applicable to all technology vendors, not just us
    - See MuleSoft's 2021 Connectivity Benchmark Report
- Cloud native engineering requires integration systems to be simply code that runs in containers
  - The days of big servers running middleware as a central service are over
- Integration needs to be simpler, less time consuming and cheaper



Integration is programming, but…

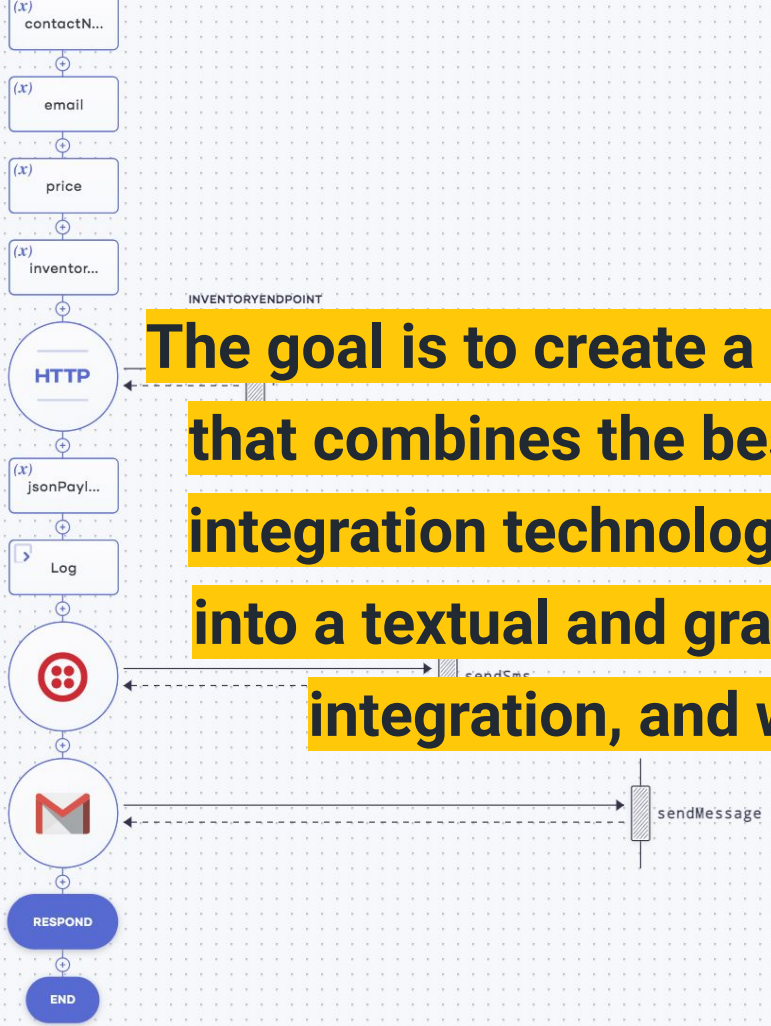
Picture credits: AltumCode on Unsplash

A visual representation of integration logic is important to communicate with business users.

Domain specific languages (DSLs) have dominated because they provide the right abstractions for integration programming, albeit with limitations when it comes to “regular code” parts of the problem.

Integration programming has lost software engineering best practices because it lives in a closed universe.

The Ballerina project



The goal is to create a modern programming language that combines the best of programming languages, integration technology, and cloud native computing into a textual and graphical language optimized for integration, and with mainstream potential

Source Edit Copy

```
5   import ballerina/http;
6
7   service on new http:Listener(8090) {
8       resource function post orders(http:Caller call
9
                                                    nic req.getJson
                                                    erPayload?.orde
                                                    ic orderPayload?.
                                                    rderPayload?.email
                                                    rderPayload?.price
                                                    s://inventoryservi
                                                    point = new (inver
                                                    ventoryResponse = <http:Re
                                                    = checkpanic inventoryResp
20
21          log:print(jsonPayload.toJsonString());
22
23          twilio:Client twilioEndpoint = new ({
24              accountSId: "ACd0fd3df1568d318124a621e
25              authToken: "0744c6a75dc9676574cb26db92
26              xAuthyKey: ""
```

The Ballerina project

- Started in 2016
- More than 300 person years of investment to date
- The language, its libraries and tools are all open source under Apache License

The Ballerina Programming Language

- Addresses most use cases of DSLs & scripting languages, but with the scalability and robustness of application languages
- Designed for mass usage - not an “elite” language
- Designed to support a graphical view
- Designed for the cloud
- “Batteries included” - comes with support for:
  - Package system
  - Structured documentation
  - Testing
  - Lots of libraries for network data, messaging & communication protocols

Features of Ballerina

Data oriented

```
// closed type
type Coord record {|
  float x;
  float y;
|};

Coord coord = { x: 1.0, y: 2.0 };

// nothing to do
json j = coord;

// If coord is open:
type Coord record {
  float x;
  float y;
};

// usually happens automatically
json j = coord.toJson();
```

- Object-orientation bundles data with code: wrong approach for network interaction
- Ballerina emphasizes plain data - data that is independent of any code used to process the data
- Ballerina provides objects for internal interfaces, but is not object-oriented
- Ballerina's plain data maps straightforwardly to and from JSON

Powerful features for working with data

```
type Employee record {
   string firstName;
   string lastName;
   decimal salary;
};

Employee[] employees = [
  // ...
];

Employee[] sorted =
   from var e in employees
   order by e.lastName ascending,
            e.firstName ascending
   select e;
```

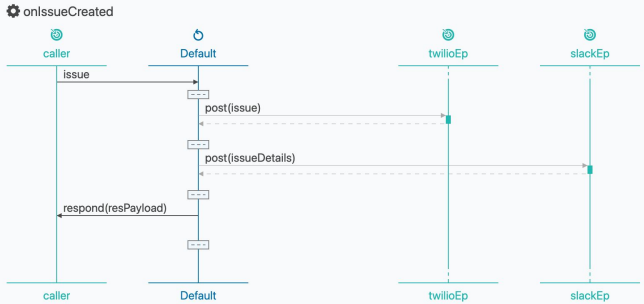
- Language integrated query with SQL-like syntax
- Table data type - work with relational and tabular data
- XML support - integrates functionality similar to XQuery
- Decimal data type - numbers designed for the needs of business

A Ballerina type system is structural and works as a schema language

- Structural, not nominal typing provides looser coupling
- Type system supports open structures: say as much or as little about the structure as you need to
- Static typing, but some things are checked at runtime in order to make the type system less complicated and more flexible
- Works for describing both the operations the program performs and data on the wire
- Works as a schema for network data as well as a type system - eliminates “data binding” problem particularly for JSON

Text and graphical syntax parity

```
remote function onIssuesCreated
      (webhook:IssuesEvent event)
                        returns error? {
  webhook:Issue issueInfo = event.issue;
  ...
```



- A function has equivalent representations as both a textual syntax and a sequence diagram
- The sequence diagram provides insight into a function's network interactions and use of concurrency
- Horizontal line for messages sent
- Only possible because it has been designed into the language from the start
  - Design of function-level concurrency features
  - Language has network abstractions

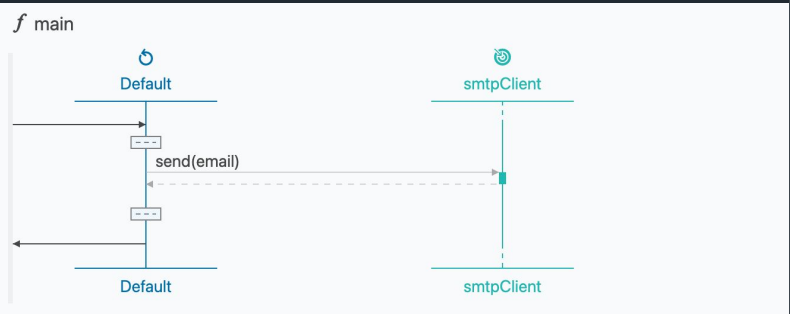
Inherently concurrent

```
function post(string message) {
   worker T {
      var r = twitter -> tweet(message);
       r -> function;
   }

   worker L {
      var r = linkedIn -> post(message);
       r -> function;
   }

   var twitterResp = <- T;
   io:println("Twitter: ", twitterResp);

   var linkedInResp = <- L;
   io:println("LinkedIn: ", linkedInResp);
}
```

- Main concurrency concept is a "strand": a thread in the concurrency sense (similar to goroutines in Go)
- A strand is cheap: does not require an OS thread
- A function can have named "workers" that each run on a new strand concurrently with the function's default worker
- Strands can be scheduled on separate OS threads to provide parallelism
- Provides the advantages of async functions with simpler programming model

Consumes network services

```
import ballerina/email;

function main() returns error? {
  email:SmtpClient sc
    = check new("smtp.example.com",
                "user123@example.com",
                "passwd123");
  check sc -> sendEmail({
     to: "contact@ballerina.io",
     subject: "Ballerina"
     body: "I love Ballerina!"
  });
}
```



- Key enabler for sequence diagram view of network interactions
- Outbound network interactions represented by client objects
- Client objects have remote methods that represent outbound interactions with a remote system
- Distinct syntax for calls remote method
- Syntax restrictions make it possible to create a sequence diagram for any function

Produces network services

```
import ballerina/http;

service / on new http:Listener(9090) {
  resource function get hello(string
                name) returns string {
    return "Hello, " + name;
  }
}
```

- Application defines service objects and attaches them to Listeners
- Libraries provide protocol-specific Listeners, which receive network input and dispatch to service objects
- Service objects support two interface styles
  - remote methods, named by verbs, support RPC style
  - resources, named by method (e.g. GET) + noun, support RESTful style (used for HTTP and GraphQL)
- Types of service objects methods can used to generate interface descriptions e.g. OpenAPI, GraphQL
- Annotations on service objects enable easy cloud deployment

Concurrency safety

- Ballerina allows strands to share mutable state in order to provide a familiar programming model
- Combination of concurrency and shared mutable state creates potential for data races (silently giving an incorrect result)
- For function workers, Ballerina avoids races by cooperatively multitasking all strands onto a single thread
- Type system has features that makes it possible to determine when services have locked enough to be able to safely use multiple threads to handle incoming requests in parallel
- Does not give massive parallelism, but enough to make effective use of common cloud instance types

Transactions

- Makes it easier to write Ballerina programs that use transactions
- Not transactional memory
- Language support for delimiting transactions
- Ballerina runtime includes transaction manager
- Composes with network interaction features to support distributed transactions

Error Handling

- Error handling approach has pervasive impact on language design and usage
- Errors are normal when you are dealing with a network
- Exceptions are the wrong approach for dealing with normal errors
  - Control flow is implicit
  - Code is harder to understand and maintain
- Trend in modern application/system languages is for error control flow to be explicit: Go, Rust, Swift
- Scripting languages typically use exceptions
- Ballerina uses error data type with explicit error control flow

Error handling example

```
configurable string host = ?;
configurable string username = ?;
configurable string password = ?;

public function main() {
   error? err = sendEmail(to = "contact@ballerina.io", subject = "Ballerina", body = "I love Ballerina");
   if err is error {
       io:println(`Error sending email: ${err.message()}`);
   } else {
       io:println("Email sent!");
   }
}

function sendEmail(string to, string subject, string body) returns error? {
   email:SmtpClient smtpClient = check new (host, username, password);
   check smtpClient->sendMessage({to, subject, body});
}
```

Ballerina is meant to be familiar

- Popular C-family languages (C, C++, Java, JavaScript, C#, TypeScript) have a lot in common
- Ballerina leverages this: there's a subset that is enough to get started and will feel very familiar to any programmer with experience of one of these C-family languages
- Ballerina provides better ways to do things, but also familiar ways to work

Ballerina offers not just the language, but the full platform

- VSCode plugin
  - Source and graphical editing
  - Debugging
- Tools for working with OpenAPI, GraphQL schemas, gRPC schemas
- Ballerina Central
  - Module sharing platform
- Integration to Choreo by WSO2 for observability, CI/CD and DevOps

Current status

Ballerina 1.0

Released in 2019

Ballerina 2021 (Swan Lake)

Major new version

Public Beta launched June 2nd 2021

Extended review / verification period before GA

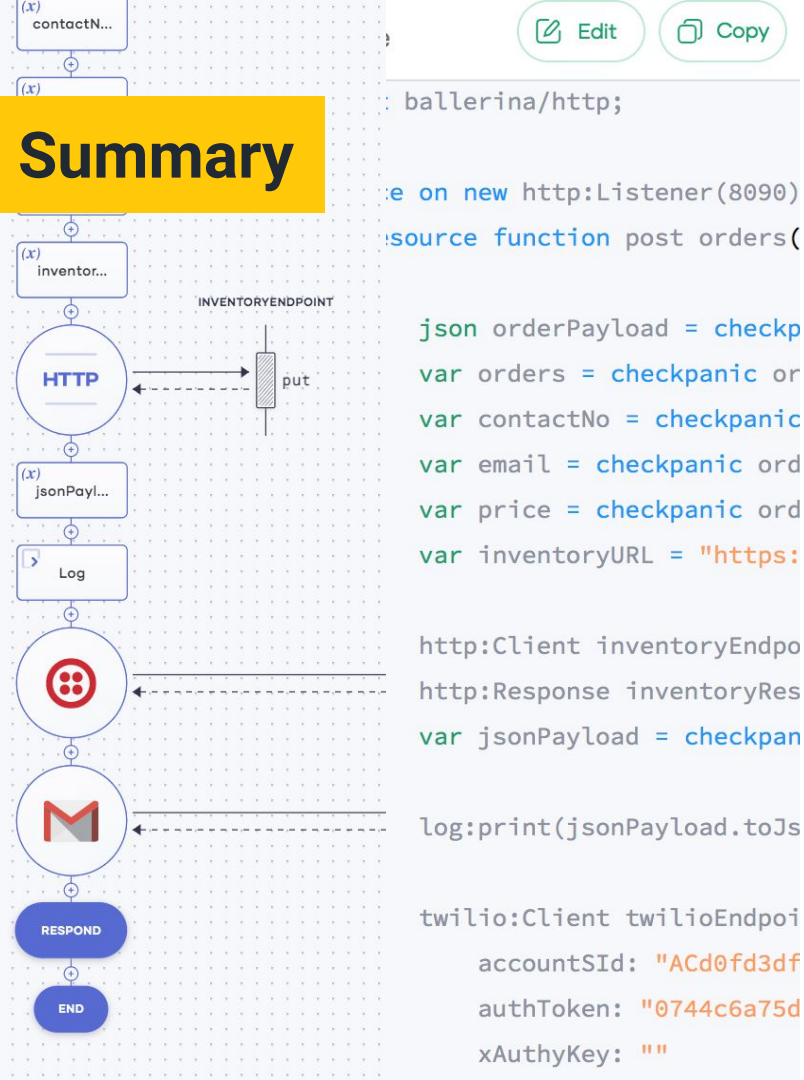
Ballerina implementations

jBallerina

- Toolchain written using Java
- Compiles to Java bytecodes and runs on a JVM
- Provides Java interoperability
- Available now

nBallerina

- Cross compilation to native binaries via LLVM
- Toolchain will be shared initially (compiler front-end still in Java) but fully bootstrapped soon
- Provides a C FFI
- ETA: initial release end of 2021



Summary

- Ballerina is a modern, industrial grade programming language optimized for writing integrations in a cloud native environment
- Type system is designed to make network data processing easier
- First class network services along with functions/objects
- Fully open source and developed openly
- Sponsored by WSO2

Thanks!

wso2.com