Integration projects always end up on one side or the other of the integration gap. Integration can either be done with integration products which are not agile or with general purpose programming languages that are not integration simple.

A programming language and runtime co-designed to be agile and integration simple.

Ballerina is a simple programming language whose syntax and runtime address the hard problems of integration. As a Turing complete language, Ballerina enables agility with edit, build, run cycles. Ballerina code is compiled into services that include transactions, embedded brokers and gateway runtimes.

```
@kubernetes:Deployment{
 image: "corp/microsocial",
}

@apiGateway:{
 security: "OAuth",
 transactionPerSec: 15
}

service<http> myService {

 @http:ResourceConfig {
   methods: ["POST"]
 }
 resource(caller, request) {

   endpoint twitter:client t {};
   transaction {
     t -> tweet( ... );
     caller -> respond( ... );
   }
}
```

Ballerina makes it easy to create resilient services that integrate and orchestrate transactions across distributed endpoints.

Language Features

TYPES AND ARRAYS > Any Type • Value Types • Type Casting • Type Inference • Type Conversion • Typeof • Objects and Records • Maps • Arrays • Arrays of Arrays • Vectors • Tuples • JSON • XML • Union Types • Tables

INTEGRATION FUNCTIONS > Task Timers • Task Appointments • Templating • HTTP Redirects • MIME Multipart Handling • AMQP and Asynchronous Messaging • Events • Transformers • Event Handling • Swagger/OpenAPI • gRPC and Protobuf • WebSockets • CORS Handling • Circuit Breakers • Load Balancing

LANGUAGE CAPABILITIES > Constants • Global Variables • Named Functions • Lambdas • Named and Default Inputs • Errors • Annotations • Async • Concurrency

INTEGRATION WORKFLOWS > Iterable Operations • Worker • Fork/Join • SQL Connector • Tables with SQL Connector • Transactions • Distributed Transactions • File API • HTTP Base Path and Path • Query Params • Tainted Data • Sensitive Data

CONTROL LOGIC > While • If Else • Foreach • Throw • Try/Catch/Finally • Match • Check • Elvis • Ternary

CONVERSIONS > JSON templates • JSON Literals • JSON Arrays • JSON/Struct/Map Conversion • Constrained JSON • XML • XML Namespaces • XML Literal • XML Templates • XML Attributes • JSON To XML Conversion • XML To JSON Conversion • Protobuf to Object Conversion • Object to Protobuf Conversion

API LOGIC > WebSocket Proxy Server • Passthrough • Mutual SSL • Caching • Byte I/O • Character I/O • Record I/O • Config API • Swagger and OpenAPI Generation • OAuth2 • Authentication • Distributed Tracing • Distributed Metrics

Platform Runtime and Tools

RUNTIMES > API Gateway • Message Broker • Identity Broker • Legacy Service Bridge • Transaction Coordinator

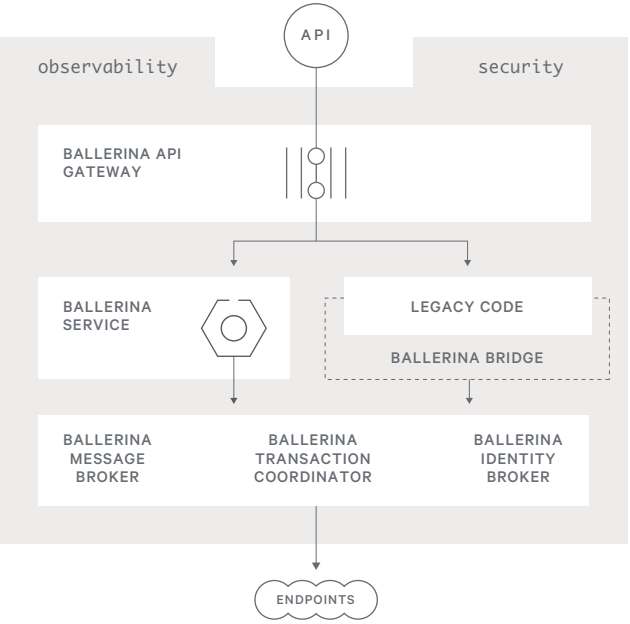
TOOLS > Build • Package • Central Package Sharing • Testerina • Composer • Documentation • VS Code • IntelliJ • Language Server • Debugger • Tracing • Docker Integration • Kubernetes Integration

Ballerina

Cloud Native Programming Language

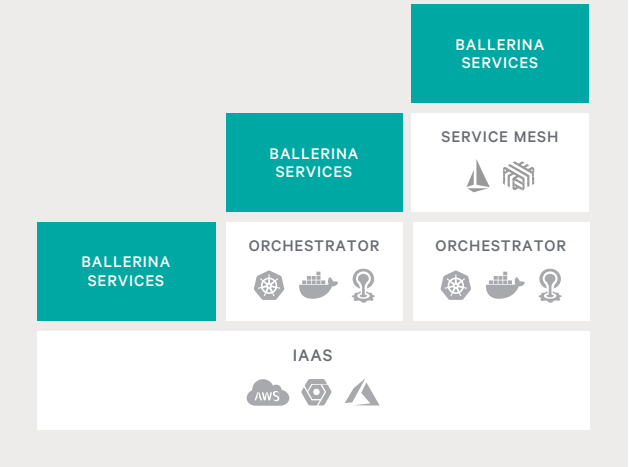
BALLERINA - CLOUD NATIVE PROGRAMMING LANGUAGE

Runtime Architecture



Developers write code, and Ballerina generates runtime and deployment artifacts with transaction resiliency built in and embedded message, security, and API brokers.

Deployment Architecture



BALLERINA KNOWS... ENDPOINTS

```
endpoint http:Client client {
 circuitBreaker: {...},
 targets: [ { url: "..."}],
};
```

BALLERINA KNOWS... PROTOCOLS

```
@http:ServiceConfig { basePath : "/" }
service<PROTOCOL> hi bind listener { }
```

BALLERINA KNOWS... JSON AND XML

```
json j = {"store": someVar, "name": anotherVar};
xml x = check j.toXML();
x.store = "Ikea";
```

BALLERINA KNOWS... DISTRIBUTED TRANSACTIONS

```
transaction { ... } failed { ... };
```

BALLERINA KNOWS... ASYNC AND THREADING

```
future<var> f1 = async postman ->
 get("/get?test=123", req);
int x = check await f1;
worker w1 = { myVar -> w2 }
worker w2 = { anothervar <- w1 }
```

BALLERINA KNOWS... TABLES AND STREAMS

```
endpoint sql:Client db { ... };
table x = check Db -> select( "SELECT * FROM TABLE");
json converted = check <json>x;

// In-memory topic, mappable to endpoints
stream<data_type> y = {};
y.publish();
y.receive();
```

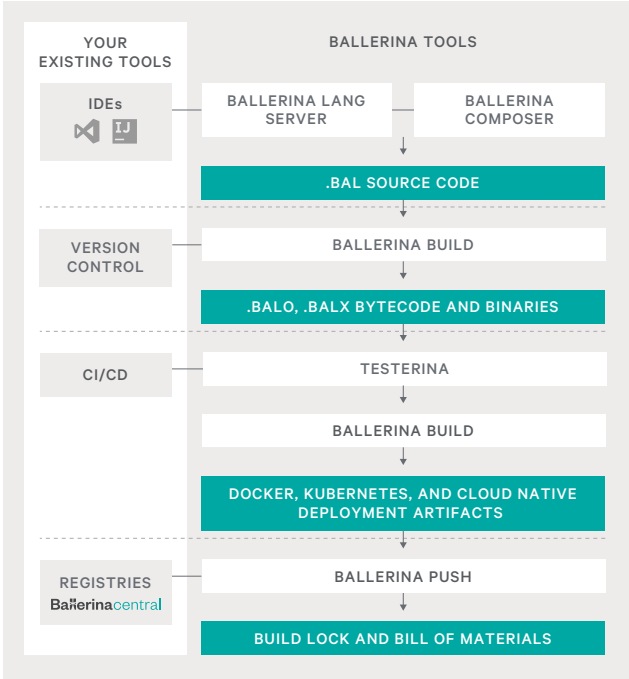
BALLERINA KNOWS... DOCKER AND KUBERNETES

```
@kubernetes:Service { }
@kubernetes:Deployment { }
```

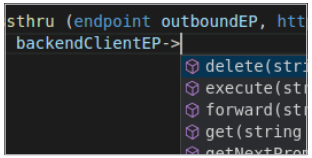
BALLERINA KNOWS... SHARED PACKAGES

```
import ballerina/http;
import ballerinax/kubernetes;
```

Lifecycle Architecture

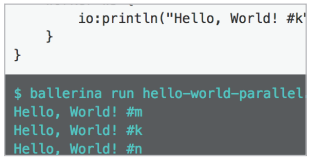


Intellisense & Debugger



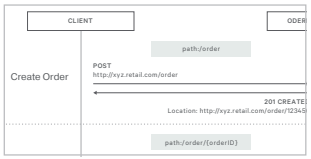
Language server with debugger, auto-completion, find references, and refactoring.

Build Management



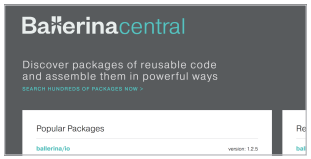
Project, package, dependency, and build management. Testerina for unit tests.

Ballerina Composer



Implement and trace services with GUI IDE for programming sequence diagrams.

Packages & Registry



Discover packages of resuable integration code and assemble them in powerful ways.