Ballerina
Swan Lake

Code to Cloud: Mastering Kubernetes Deployments with Ballerina

Conf42 Kube Native 2024

Ballerina

Anupama Pathirage
Director/ Head of Engineering - Integration
@
WSO2

anupama@wso2.com

@anupama_pathira


CLOUD
CLOUD
CLOUD INFRISTRONTIONS
CLOUD
CLOUD
CLOUD

The Journey from Code to Cloud

- The rise of cloud-native applications has transformed how we build and deploy software.
- Modern applications require seamless integration, rapid scalability, and reliable deployment strategies.
- Kubernetes has emerged as the de facto standard for orchestrating these complex, cloud-native deployments.

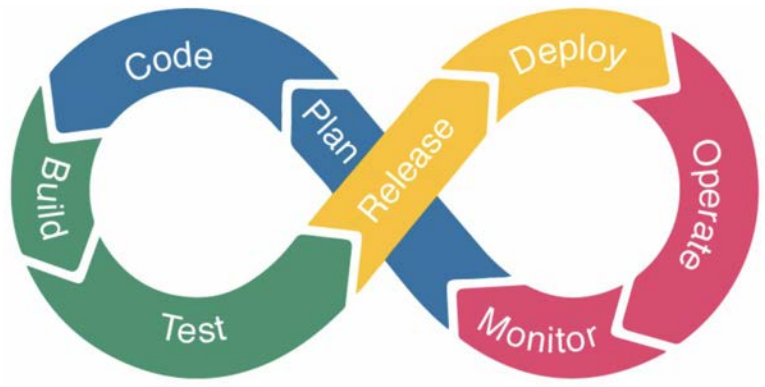


Image source : https://blog.hubspot.com/website/devops-pipeline

Challenges in Cloud Integration

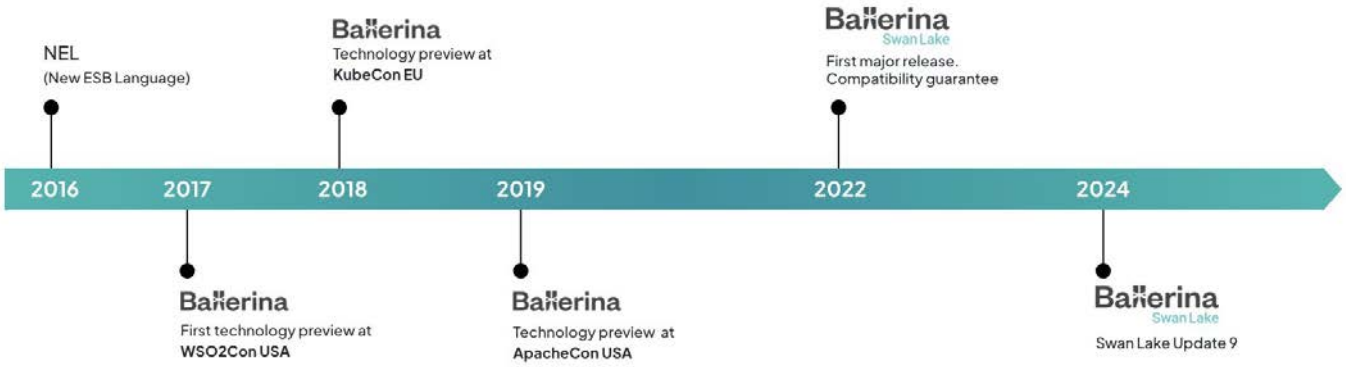
- **Multiple Services:** Cloud integration often involves numerous services and APIs that must work together.
- **Dynamic Environments:** Cloud environments are highly dynamic, with containers and microservices adding layers of complexity.
- **Configuration Overhead:** Managing deployment configurations, scaling, and monitoring can be intricate and time-consuming.
- **Security and Compliance:** Ensuring secure data flow and compliance across various cloud components is challenging.

Ballerina : A technology for simplifying cloud integrations....

Ballerina
Swan Lake

What is Ballerina?

- Open source, cloud-native technology **optimized for integration**
- **Developed by WSO2** since 2016 and first released in February 2022
- **Rich ecosystem** of different services, data formats, and connectors
- Edit/view source code **textually or graphically** as sequence diagrams and flowcharts
- Built-in, easy and **efficient concurrency** with sequence diagrams and safety primitives



Ballerina is about **cloud-native integrations**

- Ballerina: More than just a programming language; it's a full framework.
- Other Languages:
  - Many languages exist (C, C++, C#, Python, Java, Go, etc.).
  - These are general-purpose and not specifically focused on integrations or API creation.
- Importance of Abstractions:
  - Right level of abstraction, natural to the problem, is crucial.
- Ballerina provides clear and clean abstractions/tools for integration.
- Built on existing technologies and not a research product.

Ballerina provides right abstractions for …

Data

- Representing the data - shape of data, separation from code (behavior)
- Manipulating data
- Communicating data

Network

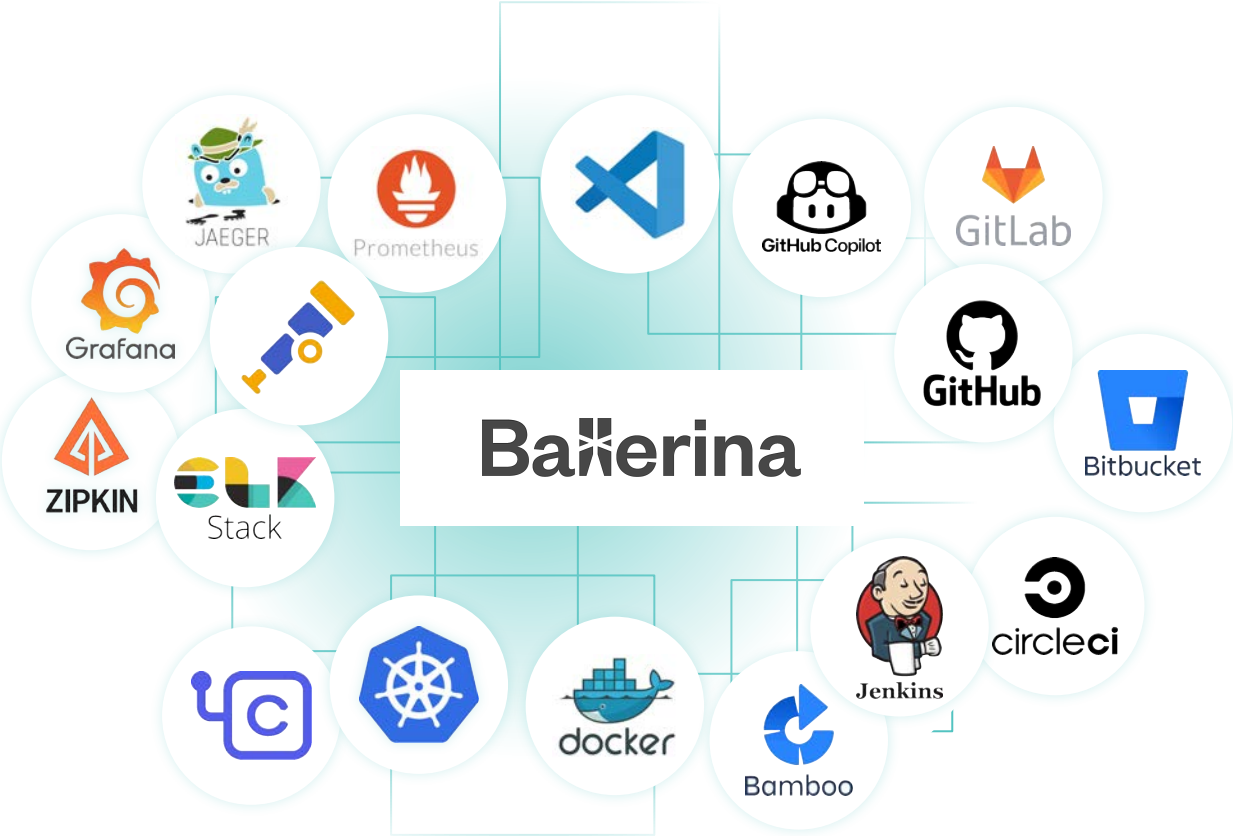
- Communicating data across different clients and services via different network protocols

Concurrency

- Concurrency safety during application scaling and inherently concurrent operations on data

Development / Maintenance

VS Code as tooling with familiar toolset



Write the first Ballerina package

- Use `bal new` command

```
bal new <package-path> [-t |--template <template-name> ]
```

E.g.

- `bal new hello-world` – Creates a package with main function
- `bal new hello-world -t service` – Creates a package with service template
- `bal new hello-world -t lib` – Creates a package with library template

Reference: https://ballerina.io/learn/cli-commands/#package-commands

Structure of Ballerina package

```
hello-world
├── .devcontainer.json
├── .gitignore
├── Ballerina.toml
└── main.bal

1 directory, 4 files
```

Not Ballerina Specific : Contains configuration files to configure a dev environment in a docker container

Not Ballerina Specific : specifies intentionally untracked files that Git should ignore like configs, build artifcats.

Identifies the directory as a Ballerina package. It contains all the meta information needed to build the package.

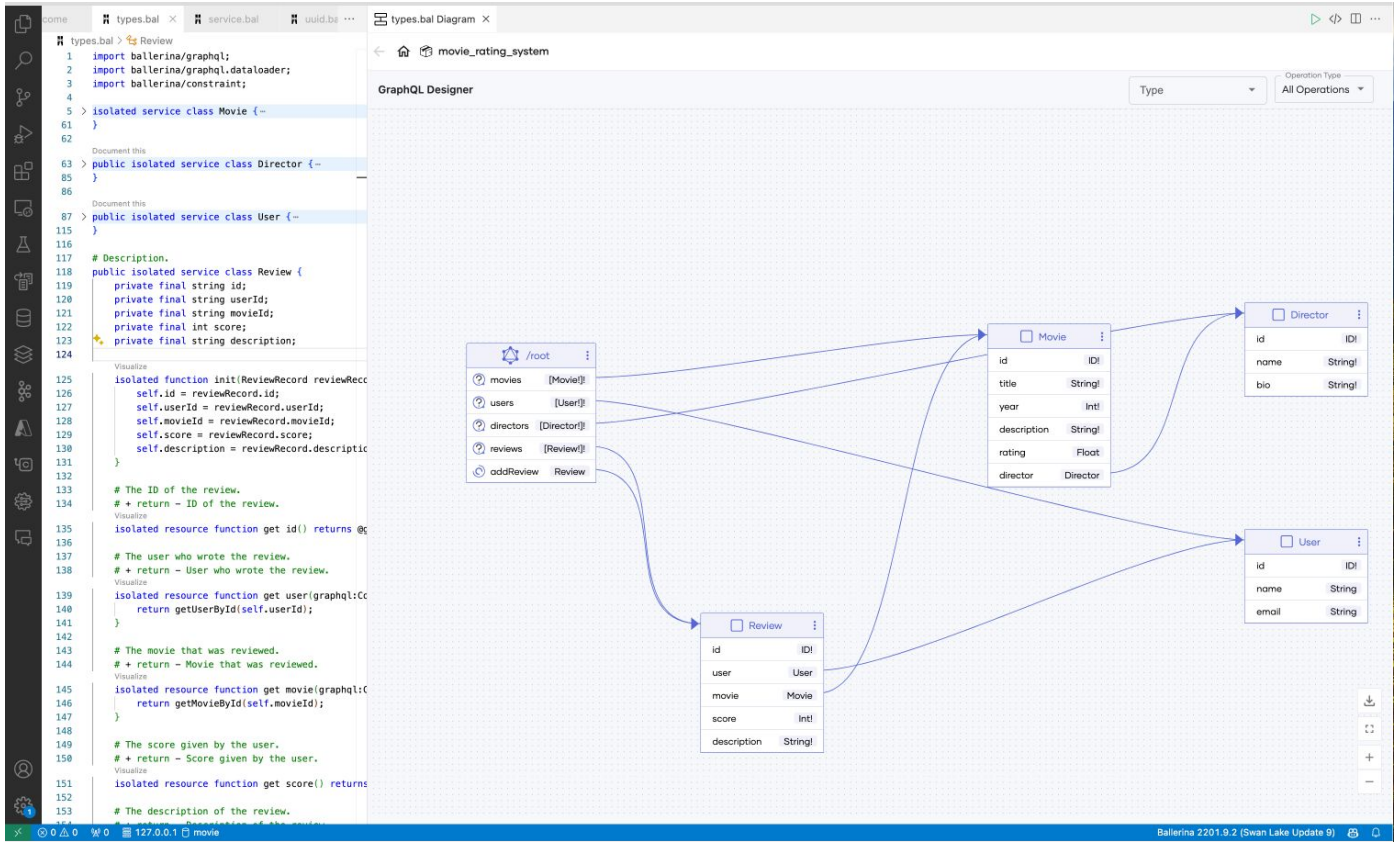
Ballerina source file.

Reference: https://ballerina.io/learn/package-references/#package-layout

Integration Designer - REST

Ballerina
Swan Lake

Integration Designer - GraphQL



Ballerina
Swan Lake

Graphical View

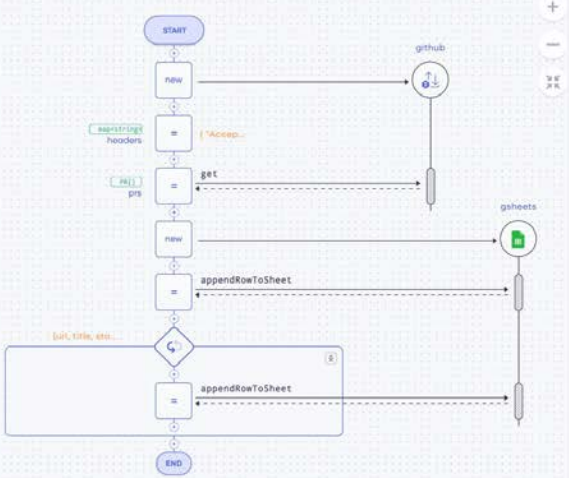
main.bal Diagram — integration-samples

main.bal

github-pull-requests-to-gsheets > main.bal > PR > url

```
import ballerina/http;
import ballerinax/googleapis.sheets;

configurable string githubPAT = ?;
configurable string sheetsAccessToken = ?;
configurable string spreadSheetId = ?;
configurable string sheetName = "Sheet1";

type PR record {
    string url;
    string title;
    string state;
    string created_at;
    string updated_at;
};

Run | Debug
public function main() returns error? {
    http:Client github = check new ("https://api.github.com");
    map<string> headers = {
        "Accept": "application/vnd.github.v3+json",
        "Authorization": "token " + githubPAT
    };

    // Network data == program data
    PR[] prs = check github->/repos/octocat/Hello\-World/pulls(headers);

    sheets:Client gsheets = check new ({auth: {token: sheetsAccessToken}});
    check gsheets->appendRowToSheet(spreadSheetId, sheetName,
            ["Issue", "Title", "State", "Created At", "Updated At"]);

    foreach var {url, title, state, created_at, updated_at} in prs {
        check gsheets->appendRowToSheet(spreadSheetId, sheetName,
                [url, title, state, created_at, updated_at]);
    }
}
```

main.bal Diagram



initial_samples* Ballerina SDK: 2201.3.2 (Swan Lake Update 3) 1 2 Live Share

Ballerina Swan Lake

In-Sync Documentation

al_media_service.bal ×

ial_media_service.bal > {} service /social\-media

```
final http:Client sentimentEndpoint = check new (se

Run | Debug | Try it | Visualize
service SocialMedia /social\-media on new http:List

    Visualize
    public function init() returns error? {
        log:printInfo("Social media service started
    }

    // Service-level error interceptors can handle
    Visualize
    public function createInterceptors() returns R
        return new ResponseErrorInterceptor();
    }

    # Get all the users
    #
    # + return - The list of users or error messag
    Visualize
    resource function get users() returns User[]|e
        stream<User, sql:Error?> userStream = socia
        return from User user in userStream
            select user;
    }
```

Swagger ×

default

GET	/users	Get all the users
POST	/users	Create a new user
GET	/users/{id}	Get a specific user
DELETE	/users/{id}	Delete a user
GET	/users/{id}/posts	Get posts for a give user
POST	/users/{id}/posts	Create a post for a given user

Schemas

Ballerina Swan Lake

Easy Data Transformations

```
30  configurable int port = 8080;
31  configurable string sheetsAccessToken = ?;
32  configurable string spreadSheetId = ?;
33  configurable string sheetName = "enrollments";
34
35  final sheets:Client gsheetsClient = check new ({auth: {token: sheetsA
36
37  const D_TIER_4_VISA = "D tier-4";
38
39  table<Person> key(id) persons = table [
40      {
41          id: 1001,
42          firstName: "John",
43          lastName: "Doe",
44          age: 25,
45          country: "LK"
46      },
47      {
48          id: 1002,
49          firstName: "Jane",
50          lastName: "Doe",
51          age: 23,
52          country: "US"
53      }
54  ];
55
56  var totalCredits = function(int total, record {string id; string name
57                          returns int => total + (course.id.startsWith("
58
    Visualize
59  function enrollPerson(Person person, Course[] courses) returns Studen
60          id: person.id.toString() + (isForeign ? "F" : ""),
61          age: person.age,
62          fullName: person.firstName + " " + person.lastName,
63          courses: from var coursesItem in courses
64              where coursesItem.id.startsWith("CS")
65              select {
66                  title: coursesItem.id + " - " + coursesItem.name,
67                  credits: coursesItem.credits
68              },
69          visaType: isForeign ? D_TIER_4_VISA : "n/a",
70          totalCredits: courses.reduce(totalCredits, 0)
71      };
72
    Run | Debug | Try it | Visualize
73  service / on new http:Listener(port) {
```

Welcome | main.bal

rest-api-with-data-mapper > main.bal > {} service / > post:persons

main.bal Diagram

rest_api_with_data_mapper enrollPerson

Data Mapper: enrollPerson

filter inpu | Auto Map | Configure

person: Person
- id: int
- firstName: string
- lastName: string
- age: int
- country: string

courses: Course[]

Local Variables
- isForeign: boolean

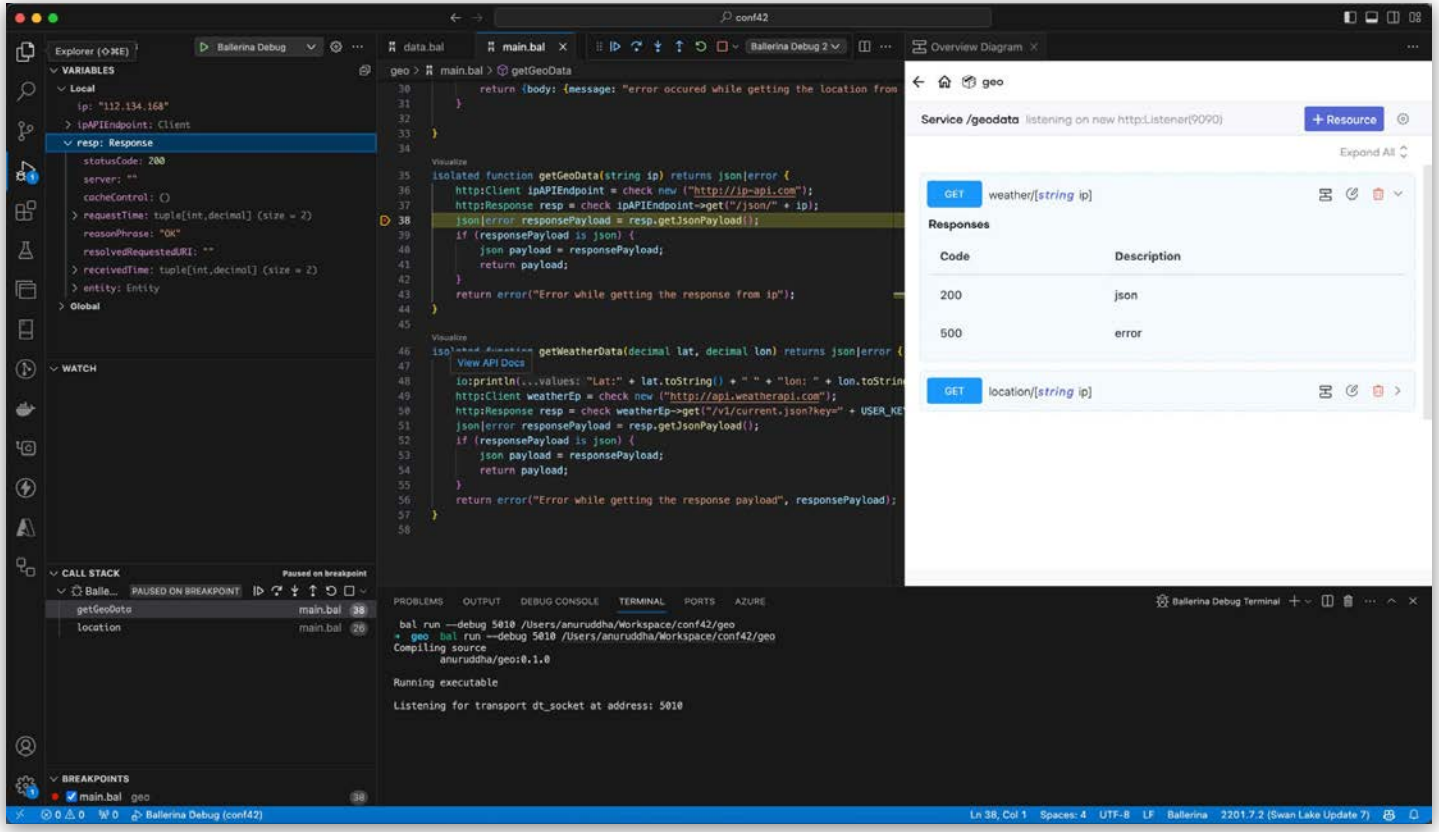
Module Variables
- D_TIER_4_VISA: string
- totalCredits: (int, course) returns int

Student
- id*: string
- fullName*: string
- age*: int
- courses*: record[]
- totalCredits*: int
- visaType*: string

0 0 | 0 | 127.0.0.1 | telco | Ballerina 2201.9.0 (Swan Lake Update 9)

Ballerina Swan Lake

Better Debugging Experience



Ballerina
Swan Lake

Everything under one roof

Ballerina Library

lang.array	2201.8.6	lang.boolean	2201.8.6	lang.decimal	2201.8.6	lang.error	2201.8.6
lang.float	2201.8.6	lang.function	2201.8.6	lang.future	2201.8.6	lang.int	2201.8.6
lang.map	2201.8.6	lang.object	2201.8.6	lang.query	2201.8.6	lang.regexp	2201.8.6
lang.runtime	2201.8.6	lang.stream	2201.8.6	lang.string	2201.8.6	lang.table	2201.8.6
lang.transaction	2201.8.6	lang.typedesc	2201.8.6	lang.value	2201.8.6	lang.xml	2201.8.6
auth	2.10.0	cache	3.7.1	cloud	2.11.3	constraint	1.5.0
crypto	2.6.2	edi	1.2.0	email	2.9.0	file	1.9.0
ftp	2.9.1	graphql	1.11.0	grpc	1.10.6	http	2.10.12
io	1.6.0	jballerina.java	2201.8.6	jballerina.java.arrays	1.4.0	jwt	2.10.0
log	2.9.0	math.vector	1.0.2	mime	2.9.0	mqtt	1.0.0
oauth2	2.10.0	observe	1.2.2	openapi	1.8.3	os	1.8.0
persist	1.2.2	protobuf	1.6.0	random	1.5.0	regex	1.3.2
soap	0.10.0	sql	1.12.0	task	2.5.0	tcp	1.9.1
test	2201.8.6	time	2.4.0	toml	0.5.1	udp	1.9.1
url	2.4.0	uuid	1.7.0	websocket	2.10.1	websub	2.10.0
websubhub	1.10.0	xmldata	2.7.0	xslt	2.6.0	yaml	0.5.3

Batteries included Ballerina connectors - 600+ Connectors

aayu.mftg.as2	1.3.1	ably	1.5.1	activecampaign	1.3.1	activemq.driver	1.0.1
adobe.analytics	1.3.1	adp.paystatements	1.5.1	adp.workerpayrollinstructions	1.5.1	alfresco	1.3.1
amadeus.flightcreateorders	1.3.1	amadeus.flightoffersprice	1.3.1	amadeus.flightofferssearch	1.3.1	Apache	0.1.0
api2pdf	1.5.1	apideck.accounting	1.5.1	apideck.lead	1.5.1	apideck.proxy	1.5.1
apple.appstore	1.5.1	asana	1.6.1	asb	3.7.0	asyncapi.native.handler	0.2.0
atspoke	1.5.1	automata	1.5.1	avatax	1.3.1	avaza	1.5.1
aws.dynamodb	2.3.0	aws.lambda	3.2.0	aws.redshift	1.0.2	aws.redshift.driver	0.1.0
aws.s3	3.3.0	aws.ses	2.1.0	aws.simpledb	2.2.0	aws.sns	2.1.0
aws.sqs	3.1.0	azure_cosmosdb	4.2.0	azure_eventhub	3.1.0	azure_storage_service	4.3.0
azure.ad	2.5.0	azure.analysisservices	1.3.1	azure.anomalydetector	1.5.1	azure.datalake	1.5.1
azure.functions	4.1.0	azure.iotcentral	1.5.1	azure.iothub	1.5.1	azure.keyvault	1.6.0
azure.openai.chat	3.0.1	azure.openai.deployment	1.0.1	azure.openai.embeddings	1.0.2	azure.openai.finetunes	1.0.1
azure.openai.text	1.0.3	azure.qnamaker	1.5.1	azure.sqldb	1.5.1	azure.textanalytics	1.5.1
azure.timeseries	1.5.1	beezup.merchant	1.6.0	bing.autosuggest	1.5.1	bintable	1.5.1
bisstats	1.5.1	bitbucket	1.5.1	bitly	1.5.1	bmc.truesightpresentationserver	1.5.1
botify	1.5.1	box	1.5.1	boxapi	0.1.1	brex.onboarding	1.5.1
brex.team	1.5.1	browshot	1.5.1	bulksms	1.5.1	candid	0.1.1
capsulecrm	1.5.1	cdata.connect	1.2.0	cdata.connect.driver	1.1.1	chaingateway	1.5.1
choreo	0.4.12	clever.data	1.5.1	client.config	1.0.1	cloudmersive.barcode	1.5.1

Ballerina
Swan Lake

Model it once, persist anywhere

Entity Relationship Diagram — ballroom-backend

model.bal M

entity_model > persist > model.bal > Submission >

```
import ballerina/time;
import ballerina/persist as _;

type User record {|
    readonly string id;
    string username;
    string fullname;
    string role;
    Challenge[] challenge;
    Contest[] moderatedContests;
    Submission[] submissions;
|};

type Contest record {|
    readonly string id;
    string title;
    byte[] readmeFile;
    time:Civil startTime;
    time:Civil endTime;
    string imageUrl;
    ChallengesOnContests[] challenges;
    Submission[] submissions;
    User moderator;
|};

type Challenge record {|
    readonly string id;
    string title;
    time:Civil createdTime;
    byte[] templateFile;
    byte[] readmeFile;
    string difficulty;
    byte[] testCasesFile;
    ChallengesOnContests[] contests;
    User author;
    Submission[] submissions;
|};
```

Entity Relationship Diagram

Entity Relationship Diagram

Collapse

Challenge

id	string
title	string
createdTime	Civil
templateFile	byte[]
readmeFile	byte[]
difficulty	string
testCasesFile	byte[]
contests	ChallengesOnContests[]
author	User
submissions	Submission[]

ChallengesOnContests

id	string
challenge	Challenge
contest	Contest
assignedTime	Civil

Contest

id	string
title	string
readmeFile	byte[]
startTime	Civil
endTime	Civil
imageUrl	string
challenges	ChallengesOnContests[]
submissions	Submission[]
moderator	User

Submission

id	string
submittedTime	Civil
score	float
challenge	Challenge
contest	Contest
user	User

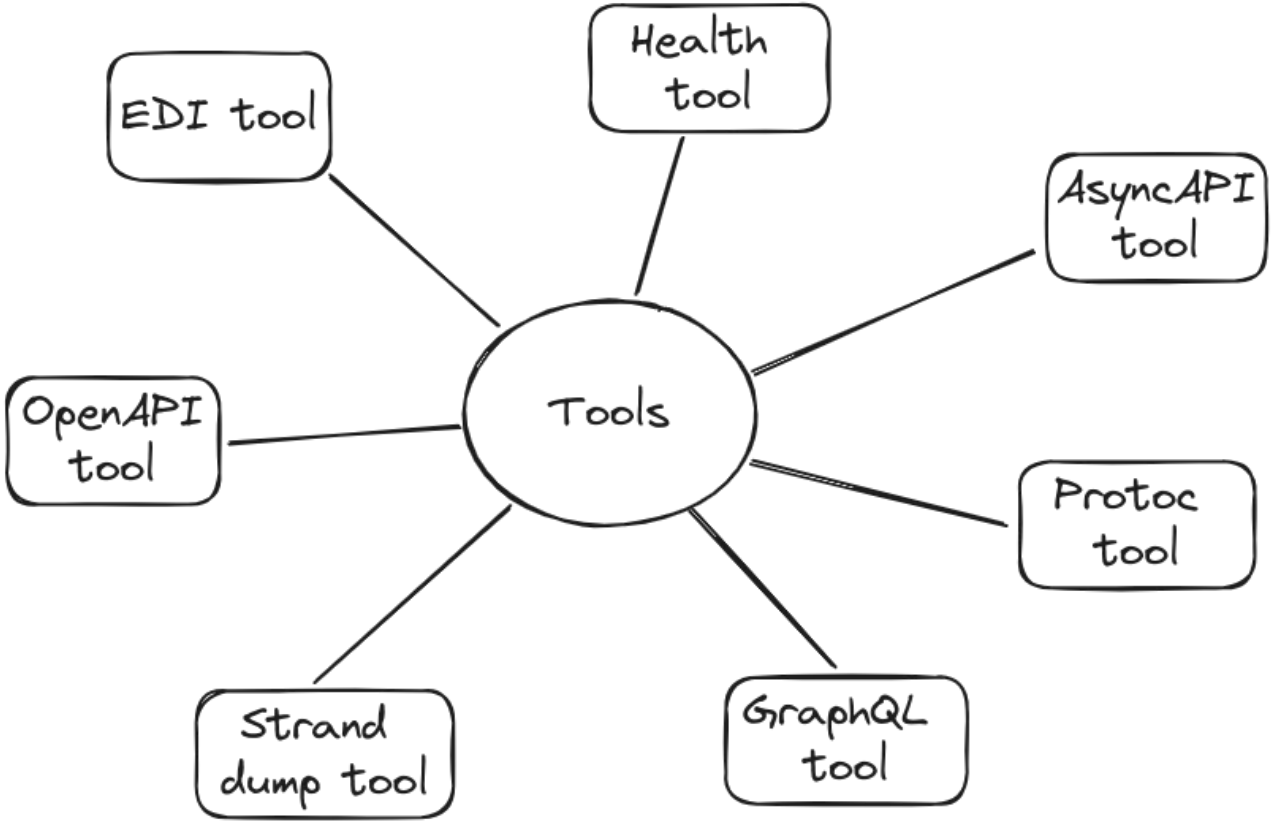
User

id	string
username	string
fullname	string
role	string
challenge	Challenge[]
moderatedContests	Contest[]
submissions	Submission[]

m2* 1 1 Live Share Ballerina 2201.7.1 (Swan Lake Update 7)

Ballerina Swan Lake

Powerful Integration Tools



Better Tests

Ballerina Test Report

socialmediasvc

1
Total tests

Passed 1/1

Failed 0/1

Skipped 0/1

Code Coverage: 34%

34%

Module Name	Total Functions				Lines Covered	Lines Missed	Coverage Percentage
socialmediasvc	1	1	0	0	34	66	34%

In-built AI Assistant

```
# Delete a user
#
# + id - The user ID of the user to be deleted
# + return - The success message or error message
Visualize
resource function delete users/[int id]() returns http:NoContent|error {
    _ = check socialMediaDb->execute(`DELETE FROM users WHERE id = ${id}`);
    return http:NO_CONTENT;
}
```

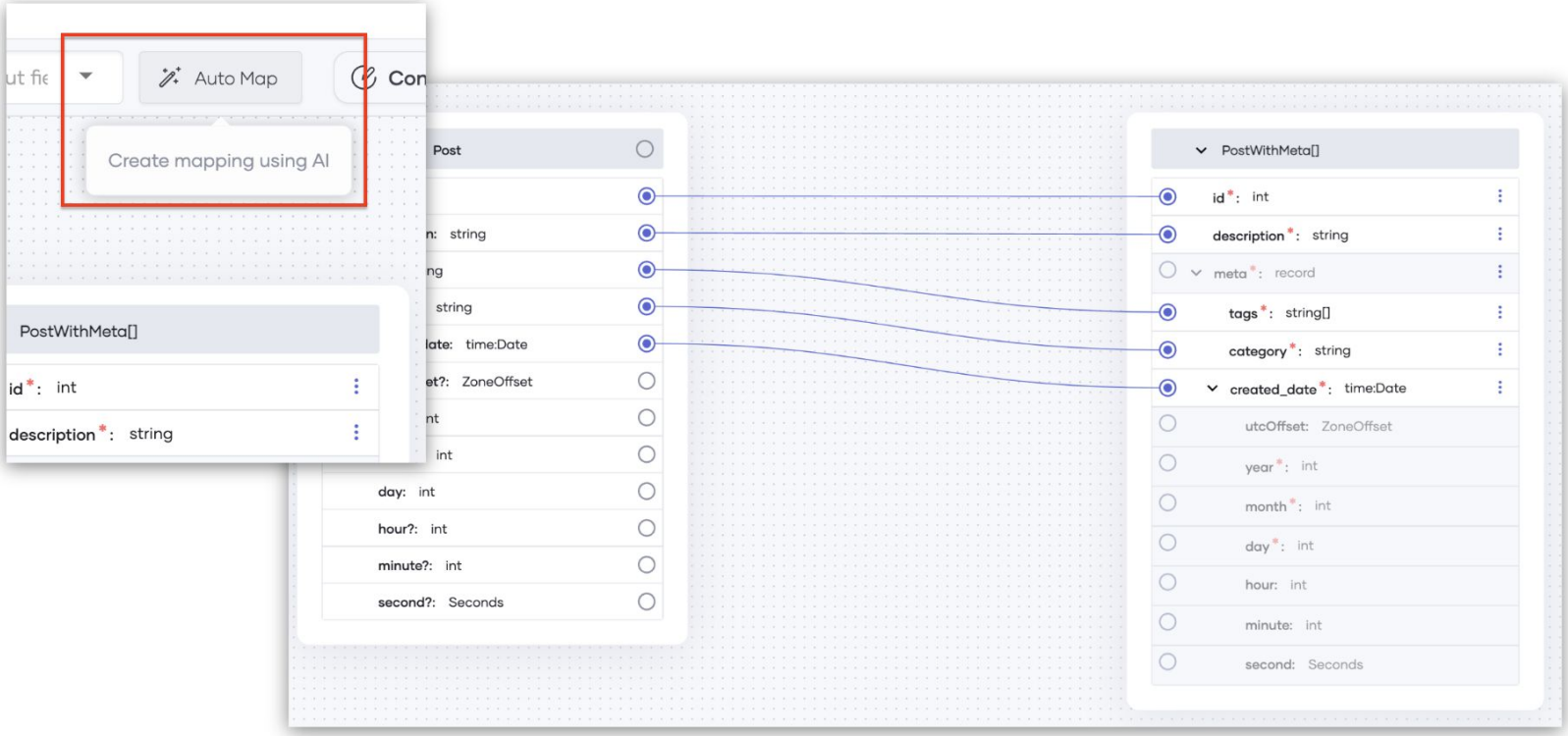
In-built AI Assistant

```
final http:Client sentimentEndpoint
Run | Debug | Try it | Visualize
More Actions...
Generate unit tests with copilot
Document this
    log:printInfo("Social media
}

// Service-level error intercept
Visualize
public function createIntercepto
```

```
ocument this
etest:Config{}
isualize
ublic function testSentimentAnalysis() returns error? {
    User userExpected = { id: 999, name: "foo", birthDate: {year: 0
    test:prepare(socialMediaDb).when("queryRow").thenReturn(userExp

    http:Client socialMediaEndpoint = check new("localhost:9095/soc
    User userActual = check socialMediaEndpoint->/users/[userExpect

    test:assertEquals(userActual, userExpected);
}
```

Ballerina
Swan Lake

In-built AI Assistant



Ballerina Swan Lake

Deployment

Build a self-contained executable (.jar)

```
→  datahandle bal build
Compiling source
        anuruddha/datahandle:0.1.0

Generating executable
        target/bin/datahandle.jar
```

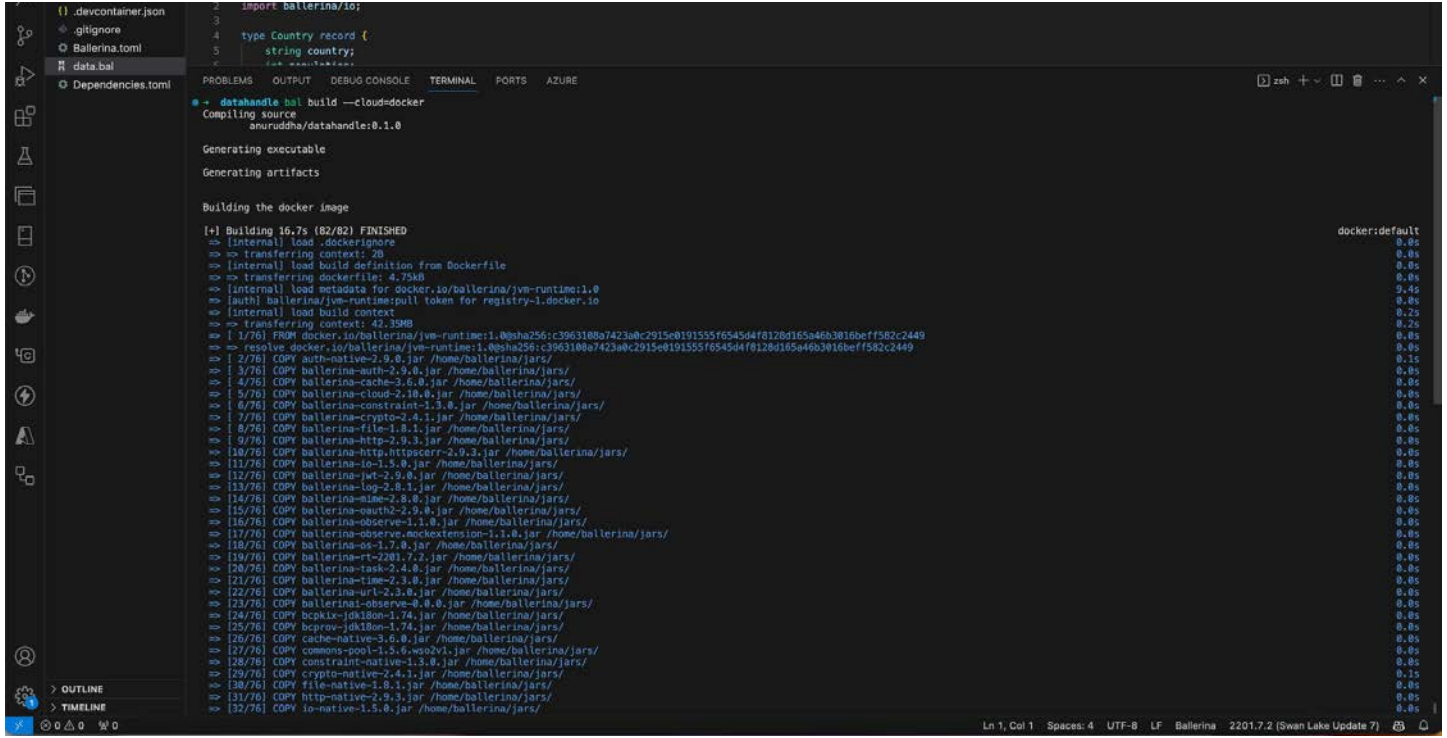
```
→  geo bal run
Compiling source
        anuruddha/geo:0.1.0

Running executable
```

Ballerina Code to Cloud is designed to allow developers to write code without thinking about the deployment platform.

This greatly simplifies the experience of developing and deploying Ballerina code in the cloud. It also enables using cloud-native technologies easily without in-depth knowledge.

Build a Docker image

```
bal build --cloud=docker
```



Work with Kubernetes

```
bal build --cloud=k8s
```

```
import ballerina/http;

service / on new http:Listener(9090) {

    // This function responds with `string` value `Hello, World!` to HTTP GET requests.
    resource function get greeting() returns string {
        return "Hello, World!";
    }
}
```

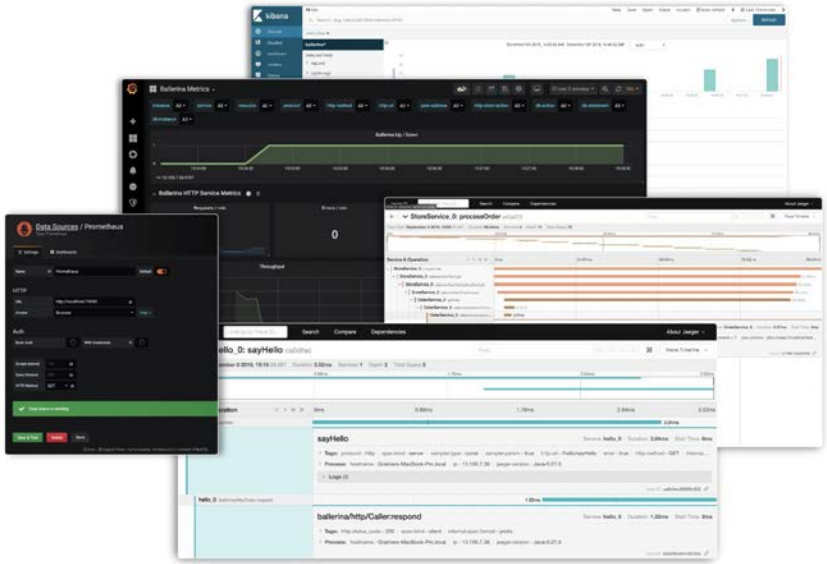
```
$ bal build --cloud=k8s
Compiling source
        ballerina/helloworld:0.1.0

Generating executable

Generating artifacts...

        @kubernetes:Service                     - complete 1/1
        @kubernetes:Deployment                  - complete 1/1
        @kubernetes:HPA                         - complete 1/1
        @kubernetes:Docker                      - complete 2/2

        Execute the below command to deploy the Kubernetes artifacts:
        kubectl apply -f /Volumes/data/ballerina/code/testBalProject/target/kubernetes/helloworld

        Execute the below command to access service via NodePort:
        kubectl expose deployment helloworld-deployment --type=NodePort --name=helloworld-svc-local

        target/bin/helloworld.jar
```

```yaml
---
apiVersion: "v1"
kind: "Service"
metadata:
  labels:
    app: "testhello"
  name: "testhello-svc"
spec:
  ports:
  - name: "port-1-testhell"
    port: 9090
    protocol: "TCP"
    targetPort: 9090
  selector:
    app: "testhello"
  type: "ClusterIP"
---
apiVersion: "apps/v1"
kind: "Deployment"
metadata:
  labels:
    app: "testhello"
  name: "testhello-deployment"
spec:
  replicas: 1
  selector:
    matchLabels:
      app: "testhello"
  template:
    metadata:
      labels:
        app: "testhello"
    spec:
      containers:
      - image: "testhello:latest"
        lifecycle:
          preStop:
            exec:
              command:
              - "sleep"
              - "15"
        name: "testhello-deployment"
        ports:
        - containerPort: 9090
          name: "port-1-testhell"
          protocol: "TCP"
```

Ballerina
Swan Lake

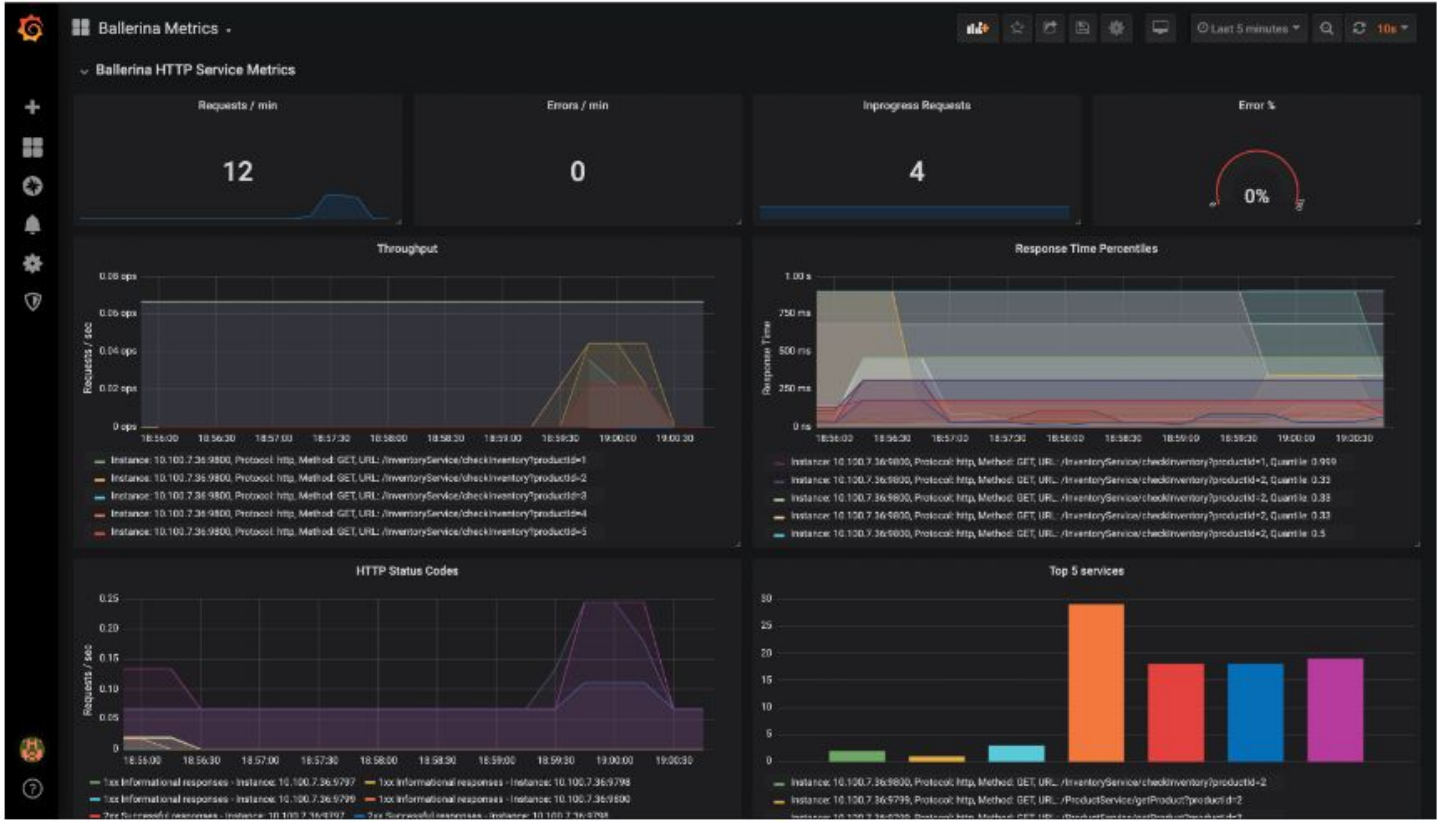
Operation

Observability in Ballerina



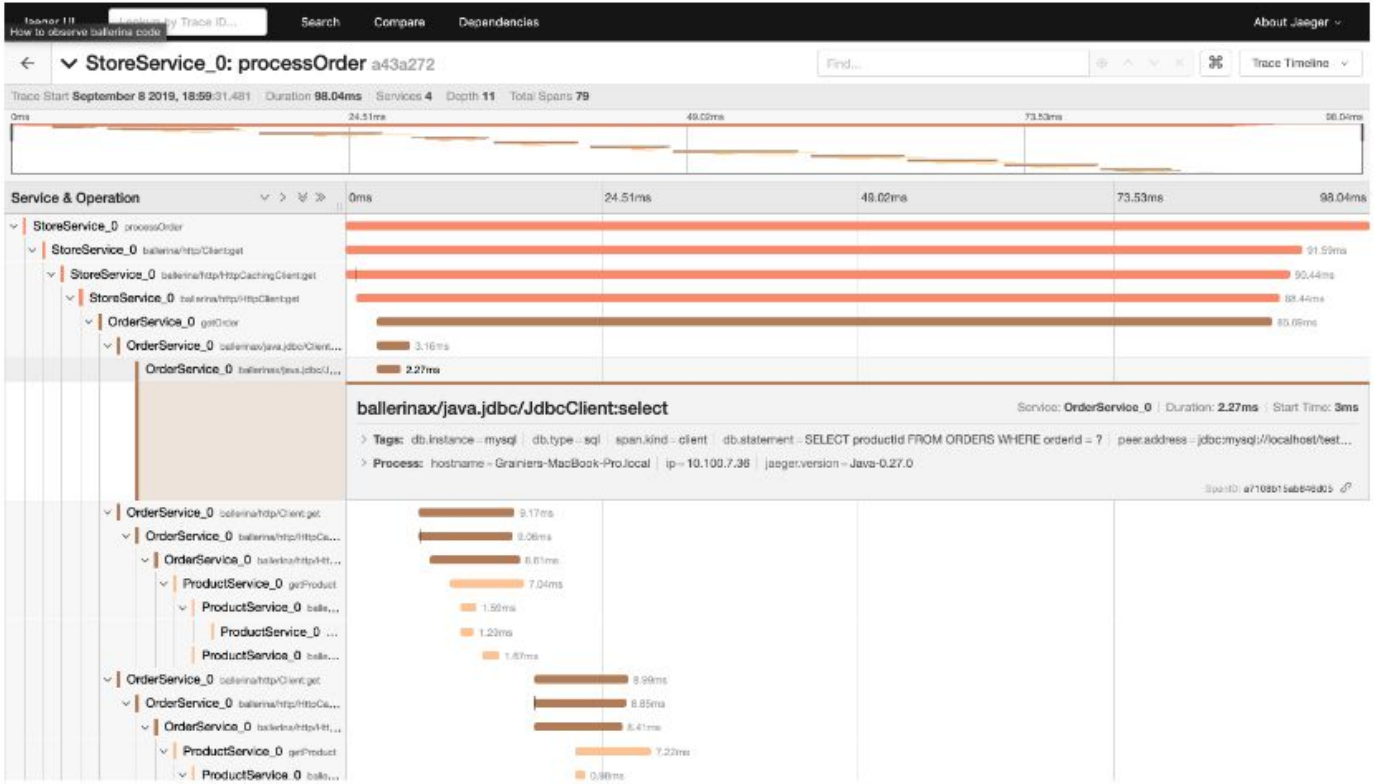
- Every Ballerina program is automatically observable by any Open Telemetry tool.
- Gives the complete control and visibility into the code's behavior and performance.
- It has 3 main pillars:
  - Metrics - Prometheus, Grafana
  - Tracing - Jaeger, Zipkin
  - Logging - Elastic Stack

Ballerina Metrics with Grafana



Ballerina
Swan Lake

Ballerina Tracing with Jaeger



Ballerina
Swan Lake

Distributed Logging

Distributed logging

In Ballerina, distributed logging and analysis are supported by the Elastic Stack. Ballerina has a log module for logging into the console. To monitor the logs, the Ballerina standard output needs to be redirected to a file.

This can be done by running the Ballerina service as below.

```
$ nohup bal run hello_world_service.bal > ballerina.log &
```

You can view the logs with the command below.

```
$ tail -f ~/wso2-ballerina/workspace/ballerina.log
```

Set up the external systems for log analytics

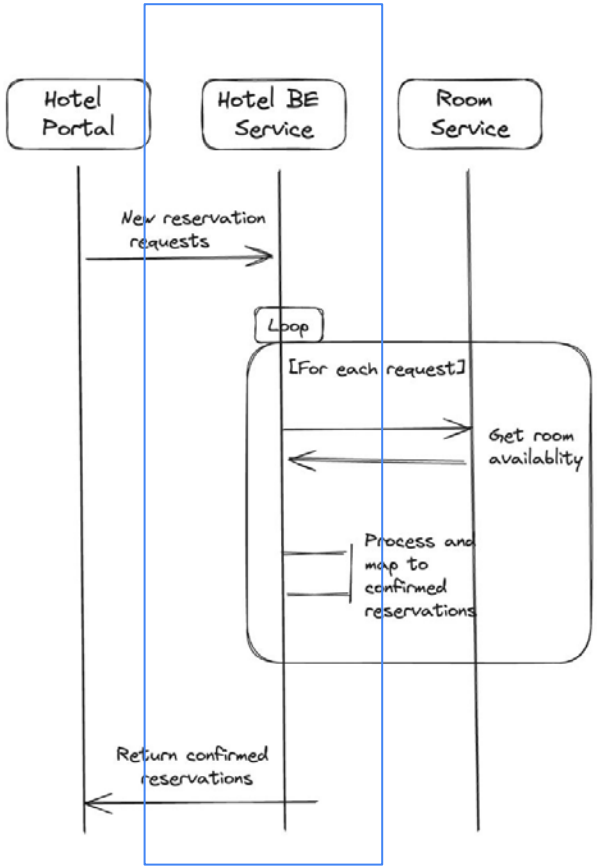
Set up Elastic Stack

The Elastic Stack comprises the following components.

1. Beats - Multiple agents that ship data to Logstash or Elasticsearch. In our context, Filebeat will ship the Ballerina logs to Logstash. Filebeat should be a container running on the same host as the Ballerina service. This is so that the log file (ballerina.log) can be mounted to the Filebeat container.
2. Logstash - Used to process and structure the log files received from Filebeat and send them to Elasticsearch.
3. Elasticsearch - Storage and indexing of the logs sent by Logstash.
4. Kibana - Visualizes the data stored in Elasticsearch.

Demo

Hotel Management Service - REST Service



Ballerina
Swan Lake

Ballerina Community

Ballerina
Swan Lake

Visit : https://ballerina.io/community/

Ballerina
Swan Lake

Visit : https://ballerina.io/community/

Find out more...

- Learn Ballerina
  - Learn pages
  - Ballerina by example
  - Ballerina VS Code extension
  - Ballerina training video series
  - Ballerina certification

- Join the Ballerina community

Discord

ballerinalang

stackoverflow

Tag : ballerina

@ballerinalang

GitHub

ballerina-lang

Thank you!

If you have any further questions, please email **contact@ballerina.io** or raise them in the **Ballerina Discord server.**

Ballerina
Swan Lake