



Hewlett Packard
Enterprise

HPE EZMERAL JAVA MICROSERVICE FRAMEWORK (EMDM) - TECH TALK



Tushar Otia
Entity MDM
Global IT – Data And Analytics

May 2022



AGENDA

1	WHY EZMERAL MICROSERVICE FRAMEWORK?	7	SECURITY BECOMES FIRST CLASS CITIZEN
2	CODE QUALITY CHECKS	8	AUTOMATIC AUDITING
3	LOGGING CONFIGURATION	9	GLOBAL EXCEPTION HANDLING
4	API DOCUMENTATION	10	ACTUATOR END POINTS
5	KUBERNETES RESOURCE DEFINATIONS	11	NEW RELIC INTEGRATION
6	CI/CD PIPELINE	12	SEAMLESS PROPERTY UPDATES
7	BAKED IN E2E DATA MONITORING	12	BUILT-IN CONNECTORS

WHY EZMERAL MICROSERVICE FRAMEWORK?

There are three main reasons I have decided to create Ezmeral Microservice Framework

DEVELOPER PRODUCTIVITY

DEVELOPER PRODUCTIVITY

DEVELOPER PRODUCTIVITY

PROVIDES FRAMEWORK FOR BUILDING CLOUD NATIVE APPS WITH MINIMUM FUSS

HELPS MAINTAIN CONSISTENCY AND STANDARDIZATION

PROVIDES BEST DEVELOPMENT PRACTICES AND PATTERNS

REDUCES THE TIME FOR ANY SI TO GET ONBOARDED

REDUCES THE TIME TO MARKET



CODE QUALITY CHECKS



GOOGLE CHECKSTYLE



- Built in support for google check style
- Enables developers to write code that adheres to coding standards (Formatting, Naming Conventions, Java Doc, Empty Catch blocks etc.)
- Automates the check style process while running locally in IDE as well as in CI/CD scripts



JaCoCo Code Coverage



- Gives the percent of your code covered by automated tests
- Gives you the cyclomatic complexity



LOGGING CONFIGURATION

Pre-built logging configuration to Console and Rolling File Appender

Rolling Policy is based on Max File Size (currently configured at 10 MB)

Automatically zips and archives the file. Archived files are deleted after 7 days



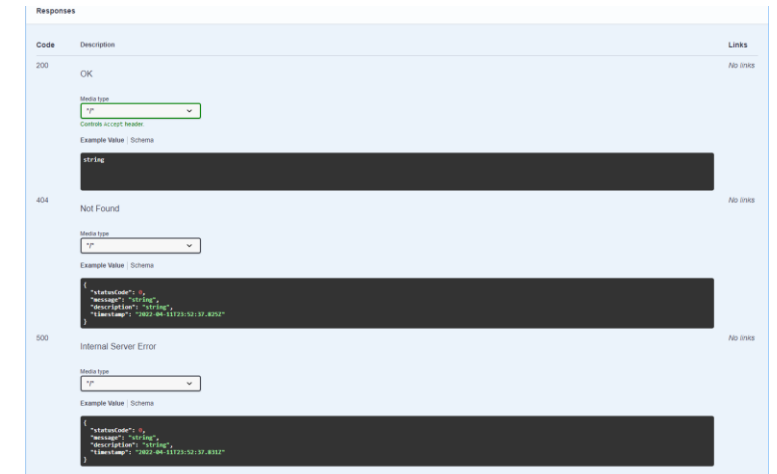
API DOCUMENTATION

Pre-built support for documenting APIs using OpenAPI 3.0
<http://localhost:xxx/your-microservice-app/v3/api-docs/>

Swagger UI is also available for API interaction and exercising any endpoints
<http://localhost:xxx/your-microservice-app/swagger-ui/index.html>

Built in support to automatically generate documentation for various HTTP response codes

When request model includes JSR-303 bean validation annotations e.g. @NotNull, @Size, @Min etc, the springdoc-openapi library uses them to generate additional schema documentation



Code	Description	Links
200	OK Media Type: <input type="text" value="application/json"/> Content-accept: Header Example Value: Schema <div>string</div>	No links
404	Not Found Media Type: <input type="text" value="application/json"/> Example Value: Schema <div>{ "statusCode": 404, "message": "Not Found", "timestamp": "2022-04-15T12:12:17.420Z" }</div>	No links
500	Internal Server Error Media Type: <input type="text" value="application/json"/> Example Value: Schema <div>{ "statusCode": 500, "message": "Internal Server Error", "timestamp": "2022-04-15T12:12:17.420Z" }</div>	No links



KUBERNETES RESOURCE DEFINATIONS

Built in customized script for running the application as docker container

Built in customized script for Gateway, Virtual Service, Destination Rule, (application) Service, Deployment, Pod definitions



CI/CD PIPELINE

Touchless deployment to HPE Ezmeral Container Platform for various environments (ITG, UAT, PROD etc.)

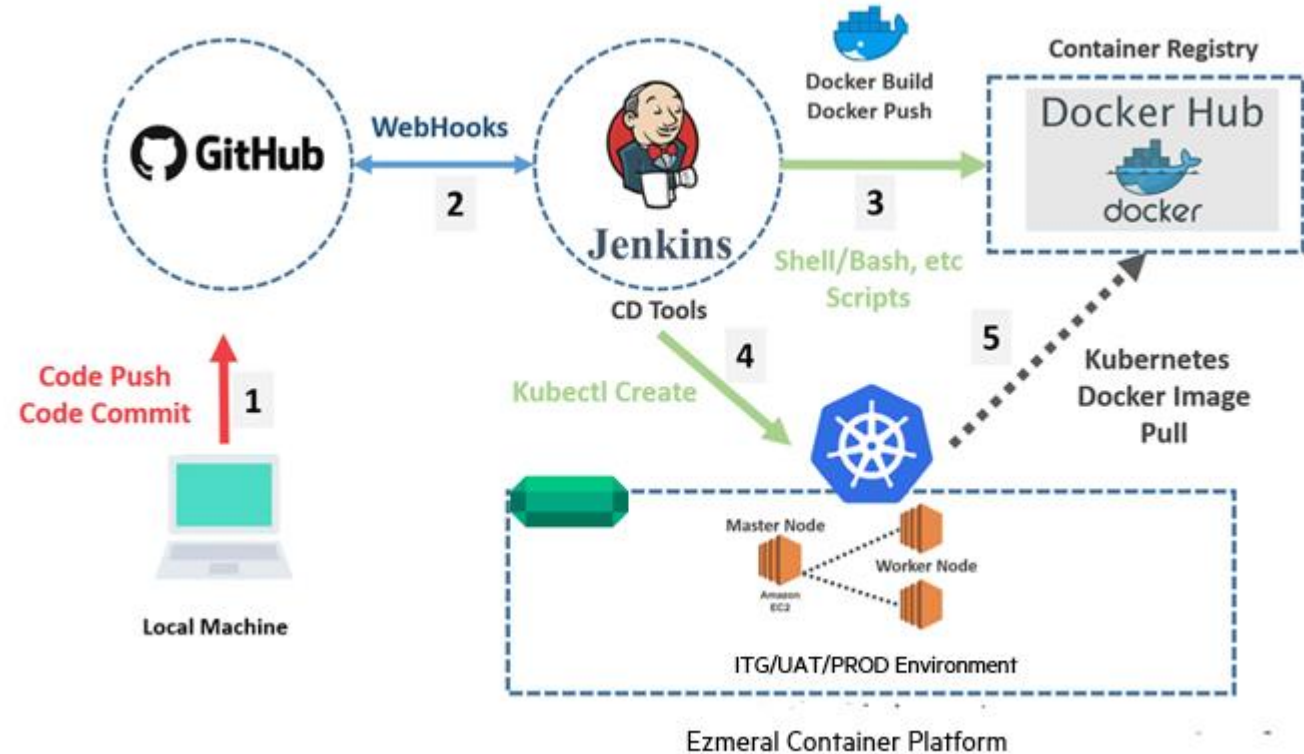
Pipeline is automatically configured based on your Organization Id, Application Name, Application Environment & Kubernetes Namespace

Convention over configuration:

<https://dns-name-itg.dc02.its.hpecorp.net/app-name/>

<https://dns-name-uat.dc02.its.hpecorp.net/app-name/>

<https://dns-name.dc01.its.hpecorp.net/app-name/> <- PROD URL



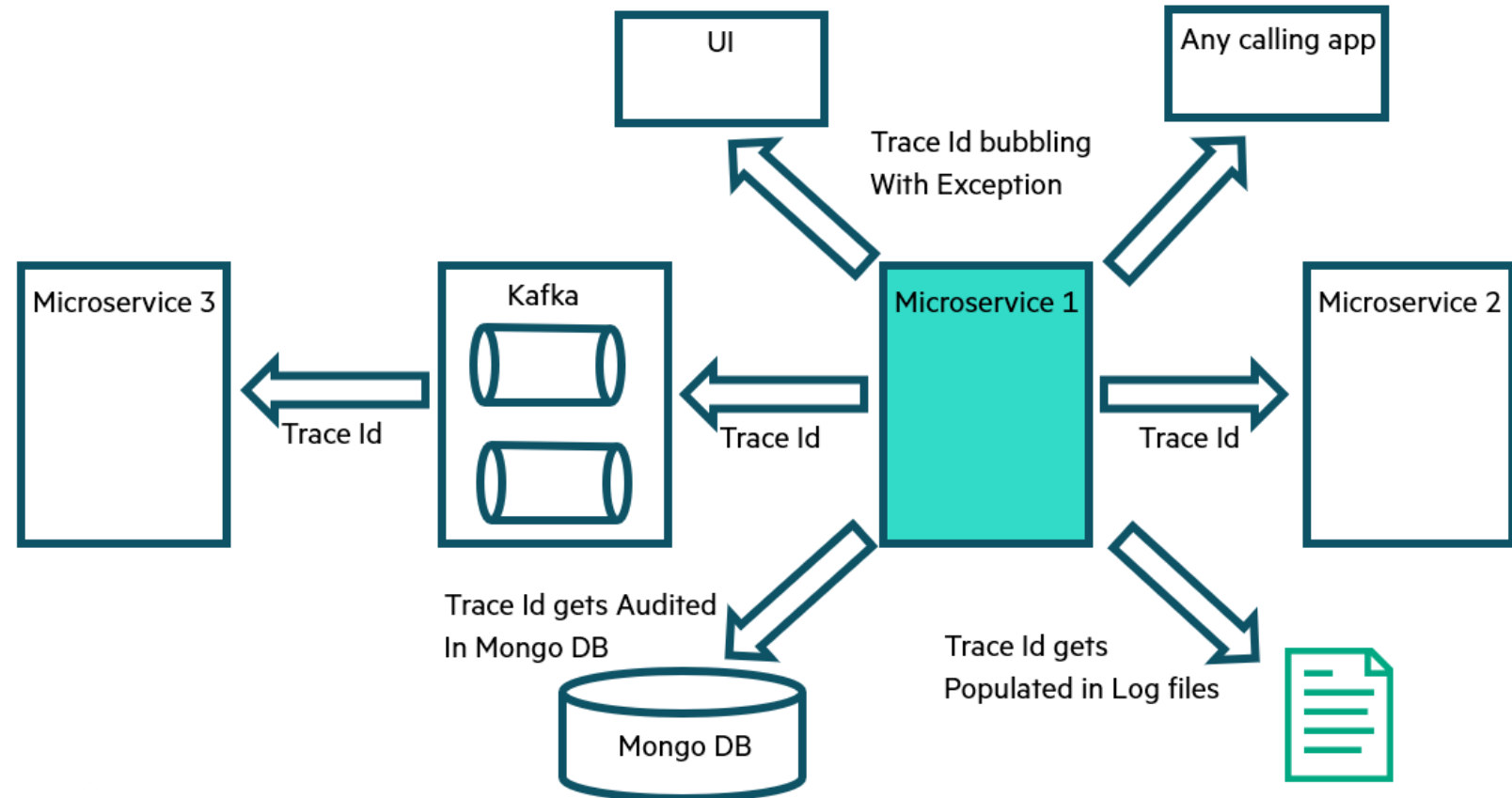
BAKED IN E2E DATA MONITORING

Weaves Trace Ids/Co-relation Ids in all the transactions automatically

Easy distributed tracing across various microservices. It will automatically inject trace ids when making a call to other microservices, Kafka calls or any exceptions that get propagated to UI or any other calling system. Trace Ids from all the microservice logs will be the same. Span Ids will be different

Automatically adds application name, trace and span ids to the Logs, so you can extract all the logs from a given trace or span in a log aggregator.

THE OMNI PRESENT TRACE ID



SECURITY BECOMES A FIRST CLASS CITIZEN

😊 No more storing passwords in GitHub with code 😊

CI/CD Integrated with Vault to fetch service account credentials

Docker Container is run as a non-root user

Framework for storing any application passwords and sensitive properties in Vault which are then automatically deployed as Kubernetes Secrets

Application Key Stores (for certificates) can be stored as Kubernetes secret (for 2 way / mutual SSL authentication)



AUTOMATIC AUDITING

Automatic auditing of logs to MongoDB

Automatically logs request body, response body, API response time, object ids (your business entity id), co-relation ids (trace ids)

Simply annotate any method with @Auditable annotation

```
@GetMapping("/")
@Auditable
public String index() {
    log.info(" Default controller called");
    return name + " service is running. The current active environment is " + activeProfile;
}
```

Object Id/Business Id can be specified as an expression language when declaring the annotation, so it will be audited along with other metrics

```
@Auditable(objectId = "#party.partyId")
public Party callSomeOtherAuditableMethod(Party
party) {
```

```
@Auditable( objectId = "#partyId")
public String index(@PathVariable
String partyId) {
```

Auditing can be turned on/off based on a property (in properties file):

```
auditing:
enable: true
```



MONGO DB

Audit Document

Application Name
API / Method Name
ObjectId / Business Id
Request Body
Response Body
Request Date
Response Time
Trace Id



GLOBAL EXCEPTION HANDLING

Uniform handling of exceptions in one central place within a microservice as well as across microservices

Consistent structure of error messages in logs as well as to any consumers e.g. UI, external services etc

Maps standard exceptions to their corresponding HTTP Status Codes.



ACTUATOR END POINTS

Automatic health check endpoint, metrics end point, environment end point, thread dumps

- <http://hostname:xxxx/app-name/actuator/health>
- <http://hostname:xxxx/app-name/actuator/metrics>
- <http://hostname:xxxx/app-name/actuator/threaddump>

Very useful to determine the health of your pods (while performing the readiness probe check)



NEW RELIC INTEGRATION



Out of the box integration with New Relic APM monitoring for your Containers

New Relic Application name will be automatically configured based on your application and environment



SEAMLESS PROPERTY UPDATES

Spring boot properties files are deployed as Config Maps in Ezmeral Container Platforms

So Properties can be updated directly in Ezmeral in the Config Maps

Perform rollout updates (without re-packaging your artifact or triggering a new deployment pipeline)

- `kubectl rollout restart deployment deployment-name` (Containers are ephemeral)

Readiness Probe ensures uninterrupted service as it makes sure that the new pods created are ready to take on requests before terminating the old pods



BUILT IN CONNECTORS

Built in connectors for

- MongoDB
- Kafka
- SFTP
- Rest Client

Connectors can be enabled or disabled based on the properties. By default they are disabled

connector:

enable:

mongo: false

kafka_producer: false

kafka_consumer: false

sftp: false



INSTRUCTIONS

```
mvn archetype:generate -DarchetypeArtifactId=hpe-ezmeral-microservice-archetype -DarchetypeGroupId=com.hpe.ngit  
-DarchetypeVersion=2.0.0
```

Please setup this profile in your local maven settings.xml file first

```
<profile>  
  <id>archetype1</id>  
  <repositories>  
    <repository>  
      <id>archetype</id>  
      <url>https://repo1.core.hpecorp.net/nexus/content/repositories/releases/</url>  
    </repository>  
  </repositories>  
</profile>
```

HELLO WORLD EXAMPLE ::

```
git@github.hpe.com:208885-entitymdm/hello-world.git
```



Q&A

THANK YOU!

