Example repo

<https://github.com/orgs/squaremiles/repositories>

* https://github.com/squaremiles/squaremiles-property
* <https://github.com/squaremiles/squaremiles-notifier>
* <https://github.com/squaremiles/helm-observability>

Docker hub:

Docker images of sqauremiles-property and squaremiles-notifier docker images are available in

* https://hub.docker.com/repositories/connectingtokabi391

Jaeger

PORT:4317

squaremiles-property

PORT:8085

squaremiles-notifier

PORT:8087

OPEN TELEMETRY

PORT: 4318

PORT:8888

Prometheus

PORT: 9090

Grafana

PORT:3000

**Maven dependencies for observability required:**

**<dependency>**

**<groupId>io.micrometer</groupId>**

**<artifactId>micrometer-tracing-bridge-otel</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>io.opentelemetry</groupId>**

**<artifactId>opentelemetry-exporter-otlp</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>io.micrometer</groupId>**

**<artifactId>micrometer-registry-otlp</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>io.micrometer</groupId>**

**<artifactId>micrometer-tracing</artifactId**

**</dependency>**

**For database tracing dependencies:**

**<dependency>**

**<groupId>net.ttddyy.observation</groupId>**

**<artifactId>datasource-micrometer-spring-boot</artifactId>**

**<version>1.0.3</version>**

**</dependency>**

Additional dependency to capture the matrices:

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-actuator</artifactId>**

**</dependency>**

**Application.properties need to be configured**

**management.endpoints.web.exposure.include=\***

**management.endpoint.health.show-details=always**

**management.otlp.metrics.export.step=10s**

**management.otlp.metrics.export.url=http://open-telemetry-opentelemetry-collector.connectingtokabi-dev.svc.cluster.local:4318/v1/metrics**

**management.otlp.tracing.endpoint=http://open-telemetry-opentelemetry-collector.connectingtokabi-dev.svc.cluster.local:4318/v1/traces**

**management.tracing.sampling.probability=1.0**

**management.tracing.enabled=true**

**logging.level.org.springframework.web.servlet.DispatcherServlet=DEBUG**

**logging.pattern.level=%5p [${spring.zipkin.service.name:${spring.application.name:}},%X{traceId:-},%X{spanId:-}]**

**management.tracing.propagation.type=w3c**

**management.metrics.tags.application=${spring.application.name}**

db matrix

**jdbc.datasource-proxy.query.enable-logging=true**

**jdbc.datasource-proxy.logging=slf4j**

**jdbc.datasource-proxy.query.log.level=DEBUG**

**jdbc.datasource-proxy.query.logger.name=square-mile-logger**

**jdbc.datasource-proxy.multiline=false**

**jdbc.datasource-proxy.include-parameter-values=true**

**Configuration for obeservility:**

**@Bean**

**public ObservedAspect observedAspect(ObservationRegistry observationRegistry) {**

**observationRegistry.observationConfig().observationHandler(new LogObservationHandler());**

**return new ObservedAspect(observationRegistry);**

}

Once configuration configured we can annotate the controller service and repo layer by

Ex:

**@Observed(name = "post.property.controller" )**

**Create and push docker image into docker hub**

**Maven settings to push the image to docker hub**

<servers> <server>

<id>registry.hub.docker.com</id>

<username>XYZ </username> <password>password </password>

</server><servers>

**Plugin**

**<plugin>**

**<groupId>com.google.cloud.tools</groupId>**

**<artifactId>jib-maven-plugin</artifactId>**

**<version>3.2.1</version>**

**<configuration>**

**<from>**

**<image>openjdk:17-oracle</image>**

**<platforms>**

**<platform>**

**<architecture>amd64</architecture>**

**<os>linux</os>**

**</platform>**

**<platform>**

**<architecture>arm64</architecture>**

**<os>linux</os>**

**</platform>**

**</platforms>**

**</from>**

**<to>**

**<image>connectingtokabi391/squaremiles</image>**

**<tags>**

**<tag>1.0.5</tag>**

**</tags>**

**</to>**

**<containerizingMode>packaged</containerizingMode>**

**<container>**

**<jvmFlags>**

**<jvmFlag>-Xms512m</jvmFlag>**

**<jvmFlag>-Xdebug</jvmFlag>**

**</jvmFlags>**

**<ports>**

**<port>8085</port>**

**</ports>**

**<format>Docker</format>**

**</container>**

**</configuration>**

**<executions>**

**<execution>**

**<id>package</id>**

**<phase>package</phase>**

**<goals>**

**<goal>build</goal>**

**</goals>**

**</execution>**

**</executions>**

**</plugin>**

**Step to deploy application into redhat openshift**

[**https://oauth-openshift.apps.sandbox-m4.g2pi.p1.openshiftapps.com/oauth/token/display**](https://oauth-openshift.apps.sandbox-m4.g2pi.p1.openshiftapps.com/oauth/token/display)

**oc login --token=sha256~Ft7SBQptWuhua0a2evtsg9p7Ces2nriB1fQF\_1F823g --server=https://api.sandbox-m4.g2pi.p1.openshiftapps.com:6443**

**Step 1 :**

**Oc client need to install**

[**https://console-openshift-console.apps.sandbox-m4.g2pi.p1.openshiftapps.com/command-line-tools**](https://console-openshift-console.apps.sandbox-m4.g2pi.p1.openshiftapps.com/command-line-tools)

**Helm need to install**

[**https://helm.sh/docs/intro/install/**](https://helm.sh/docs/intro/install/)

**login to openshift:**

**Step -2:**

* **Take clone from** <https://github.com/squaremiles/helm-observability>

1. Deploy squaremiles property

helm upgrade -i helm-squaremiles-release-1 ./helm-squaremiles

1. Deploy squaremiles notifier

helm upgrade -i helm-squaremiles-notifier-release-1 ./helm-squaremiles-notifier

1. Deploy opentelemetry

cd helm-opentelemetry

helm upgrade -i open-telemetry open-telemetry/opentelemetry-collector -f ./values.yml

1. Deploy Jaeger

cd jeager

helm upgrade -i jaeger jaegertracing/jaeger --values /.values.yaml

1. Deploy Prometheus

cd helm-prometheus

helm upgrade -i -f prometheus.yaml prometheus prometheus-community/Prometheus

1. Deploy Grafana

cd helm-grafana

helm upgrade grafana grafana/grafana -f values.yml

Note 1: if the above images are not available, we need to add repos from below url

<https://github.com/open-telemetry/opentelemetry-collector>

<https://github.com/jaegertracing/jaeger>

<https://github.com/prometheus-community/helm-charts>

<https://github.com/grafana/grafana>

Note-2

: Generally for prometheous and Grafana we need to create service account and need to bind as follow

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

  name: prometheous-demo-role

  namespace: connectingtokabi-dev

  uid: ba393299-b1d2-454f-ae58-3e0a28eb0c83

  resourceVersion: '1284208623'

  creationTimestamp: '2024-01-28T04:11:58Z'

  managedFields:

    - manager: Mozilla

      operation: Update

      apiVersion: rbac.authorization.k8s.io/v1

      time: '2024-01-28T04:11:58Z'

      fieldsType: FieldsV1

      fieldsV1:

        'f:rules': {}

rules:

  - verbs:

      - get

      - watch

      - list

    apiGroups:

      - ''

    resources:

      - pods

      - namespaces

  - verbs:

      - get

      - list

      - watch

    apiGroups:

      - apps

    resources:

      - replicasets

  - verbs:

      - get

      - list

      - watch

    apiGroups:

      - extensions

    resources:

      - replicasets

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

  name: prometheous-demo-rb

  namespace: connectingtokabi-dev

  uid: 34622cc8-6826-4f25-9724-9fae349ed029

  resourceVersion: '1284211960'

  creationTimestamp: '2024-01-28T04:13:10Z'

  managedFields:

    - manager: Mozilla

      operation: Update

      apiVersion: rbac.authorization.k8s.io/v1

      time: '2024-01-28T04:13:10Z'

      fieldsType: FieldsV1

      fieldsV1:

        'f:roleRef': {}

        'f:subjects': {}

subjects:

  - kind: ServiceAccount

    name: prometheous-demo

    namespace: connectingtokabi-dev

roleRef:

  apiGroup: rbac.authorization.k8s.io

  kind: Role

  name: prometheous-demo-role