# Labo tracing

|  |  |
| --- | --- |
| 1 | si primera vez  git clone https://github.com/luitoz/curso-microservicios  Bajar fuentes  git pull  verificar que se hayan creado los objetos en sesion7/demo-tracing |
| 2 | Crearse una cuenta en papertrail  https://papertrailapp.com |
| 3 | login en https://papertrailapp.com y obtener cadena de conexion |
| 4 | configurar logspout   * docker/common/docker-compose.yml   reemplazar cadena conexion  logspout:  image: gliderlabs/logspout  command: <your-string-connection>  volumes:  - /var/run/docker.sock:/var/run/docker.sock |
| 5 | desplegar  mvn clean package docker:build && docker-compose -f docker/common/docker-compose.yml up |
| 6 | http://localhost:5555/api/licensing/v1/organizations/e254f8c-c442-4ebe-a82a-e2fc1d1ff78a/licenses/f3831f8c-c338-4ebe-a82a-e2fc1d1ff78a  verificar trace information en terminal |
| 7 | verificar y buscar logs en papertrail |
| 8 | add correlation en zuul postfilter   * *nota: hay metodo alternativo*   *https://cloud.spring.io/spring-cloud-static/spring-cloud-sleuth/1.0.12.RELEASE/#\_running\_examples* |
| 9 | http://localhost:5555/api/licensing/v1/organizations/e254f8c-c442-4ebe-a82a-e2fc1d1ff78a/licenses/f3831f8c-c338-4ebe-a82a-e2fc1d1ff78a  verificar correlation en header |
| 10 | buscar traceid en papertrail |

## Labo Integrar sleuth con zipkin

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | | demo-zipkin/  ver  zipkinsvr/ | |
| 2 | configurar clientes zipkin   * licensing-service/ * organization-service/ * zuulsvr/ | |
| 3 | tracing level   * zuulsvr/src/main/java/com/thoughtmechanix/zuulsvr/ZuulServerApplication.java * organization-service/src/main/java/com/thoughtmechanix/organization/Application.java * licensing-service/src/main/java/com/thoughtmechanix/licenses/Application.java | |
| 4 | desplegar  cd demo-zipkin  mvn clean package docker:build && docker-compose -f docker/common/docker-compose.yml up | |
| 5 | acceder  get http://localhost:5555/api/organization/v1/organizations/e254f8c-c442-4ebe-a82a-e2fc1d1ff78a/  analizar traces en http://localhost:9411 | |

## Labo Integrar Postgress con Zipkin

|  |  |
| --- | --- |
| 1 | agregar customspan de servicio postgress   * organization-service/src/main/java/com/thoughtmechanix/organization/services/OrganizationService.java   modificar  ...  @Autowired  private Tracer tracer;  ...  public Organization getOrg  (String organizationId) {  Span newSpan = tracer.createSpan("getOrgDBCall");//1  logger.debug("In the organizationService.getOrg() call");  try {  newSpan.tag("postgress.op","select");//2  newSpan.logEvent(org.springframework.cloud.sleuth.Span.CLIENT\_SEND);  return orgRepository.findById(organizationId);  }  finally{  newSpan.tag("peer.service", "postgres");//4  newSpan.logEvent(org.springframework.cloud.sleuth.Span.CLIENT\_RECV);  tracer.close(newSpan);//5  }  }  ... |
| 2 | desplegar  mvn clean package docker:build -f organization-service/pom.xml && docker-compose -f docker/common/docker-compose.yml up organizationservice |
| 3 | http://localhost:5555/api/licensing/v1/organizations/e254f8c-c442-4ebe-a82a-e2fc1d1ff78a/licenses/f3831f8c-c338-4ebe-a82a-e2fc1d1ff78a  zipkin  <http://localhost:9411/>  1. buscar por traceid  2. analizar traces |

## Labo elk stack

|  |  |
| --- | --- |
| 1 | * demo-elk |
| 2 | ver elk -config   * elk-Image/02-beats-input.conf * elk-Image/30-output.conf * elk-Image/Dockerfile |
| 3 | instalar elk  cd elk-Image  docker build . --tag local-elk  sudo sysctl -w vm.max\_map\_count=262144  docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it --name myelk local-elk |
| 4 | microservicio order-service  ver   * order-service/pom.xml * order-service/src/main/resources/application.properties * order-service/src/main/resources/logback.xml |
| 5 | ejecutar 3 microservicios  cd demo-elk  mvn spring-boot:run -f order-service/pom.xml  mvn spring-boot:run -f customer-service/pom.xml  mvn spring-boot:run -f orchestrator/pom.xml |
| 6 | invocar  http://localhost:8080/customer-orders/100 |
| 7 | visualizar traces en kibana   * acceder kibana * crear index pattern logstash-local   + agregar filtro de tiempo * buscar traces en el index * buscar con KQL |
|  |  |