> 本方案描述如何在日志中打印RequestId，进而根据RequestId，串联起单个模块中同一请求的全部日志，或者在不同模块之间进行链路追踪。
>
> 本方案以创建云主机为例，说明方案的具体实施过程。

1. 创建云主机的请求，首先打到网关，网关会生成RequestId，并存储到UserSession。

   由于UserSession会从网关开始，传递到链路中的每个节点，包括异步任务中，所以本方案选择将RequestId存储在UserSession中。

```
public class UserSession implements Serializable {
   /**
     * 请求ID，用来在日志中打印，进行同一请求的链路追踪
     */
    private String requestId;

    public String getRequestId() {
        return requestId;
    }

    public void setRequestId(String requestId) {
        this.requestId = requestId;
    }
}
```

2. 请求接着到达ICompute，由ICompute中拦截所有Controller的AOP，将UserSession中的RequestId设置为logback变量。

```
@Slf4j
@Aspect
@Component
public class RequestIdAdvice {

    @Pointcut("within(com.inspur.incloud.icompute.openstack.controller..*)")
    private void allController() {}

    @Around("allController()")
    public Object doAround(ProceedingJoinPoint joinPoint) throws Throwable {
        try {
            // 1.将requestId设置为logback变量
            MDC.put("REQUEST_ID", SessionCheckUtil.getSession().getRequestId());

            // 2.业务逻辑处理
            return joinPoint.proceed();
        } finally {
            // 3.清理logback变量
            MDC.clear();
        }
    }
}
```

3. 更新logback配置文件，取出步骤2中设置的RequestId，作为日志的一部分。

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d %p %X{REQUEST_ID} (%file:%line\)- %m%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
    </appender>
    <root level="INFO">
        <appender-ref ref="STDOUT" />
    </root>
</configuration>
```

以上配置中的 `%X{REQUEST_ID}` 是logback取MDC中设置的配置的语法。

4. ICompute继续添加ITask任务，异步创建云主机。但是步骤2中设置的RequestId，在异步任务中取不到。

   为了解决这个问题，需要对ITask中`AsynServiceImpl`类`addWork`方法进行改造，将UserSession中的RequstId再次赋值给Slf4j的MDC对象。

5. 在创建云主机的异步任务中，ICompute会调用INetwork、ICinder等模块的接口。

   这些模块也依照步骤2、3、4进行配置，从ICompute传递过来的UserSession中取出RequestId，打印到自己的模块日志中。

6. 完成上述步骤后，ICompute及请求链路上的其他模块的日志，就会带上RequestId，控制台输出示例如下：

```
2023-10-31 20:14:30,481 ERROR 8a6f34e58b7432ce018b752420ca0026
(OpenstackVmCreateServiceImpl.java:363)- vm name :zhvm02 existed
```

7. 更新项目内公共返回Model，添加RequestId并在构造器中赋值，返回给前端或经由网关的接口调用。

```
public class OperationResult<T> implements java.io.Serializable {

    private String requestId;

    public String getRequestId() {
        return requestId;
    }

    public OperationResult() {
        this.requestId = SessionCheckUtil.getSession().getRequestId();
    }
}
```

返回结果示例：

```
{
    "flag": true,
    "errCode": null,
    "errMessage": null,
    "exceptionMsg": null,
    "resData": {
        "total": 0,
        "pageSize": 10,
        "currentPage": 1,
        "totalPages": 0,
        "data": []
```

```
    },
    "requestId": "8a6f34e58b7432ce018b752420ca0026"
}
```


备注：

1. 如果希望自开启线程中打印RequestId，需要侵入到具体业务处理。