拉勾教育
—互联网人实战大学—

《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

— 拉勾教育出品 —

拉勾教育

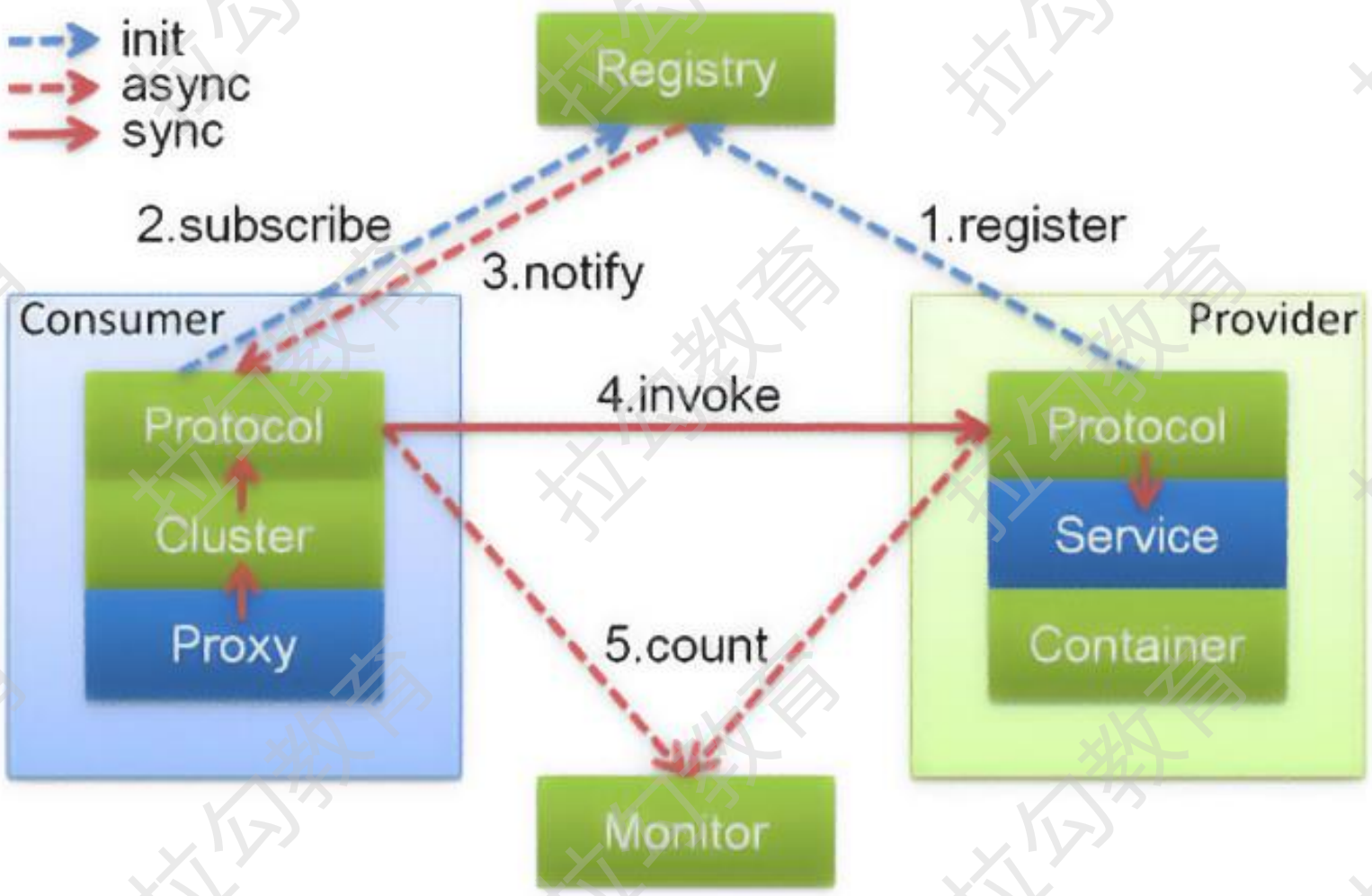
拉勾教育
—互联网人实战大学—

第17讲：Dubbo 插件核心剖析 Trace 是这样跨服务传播的

拉勾教育

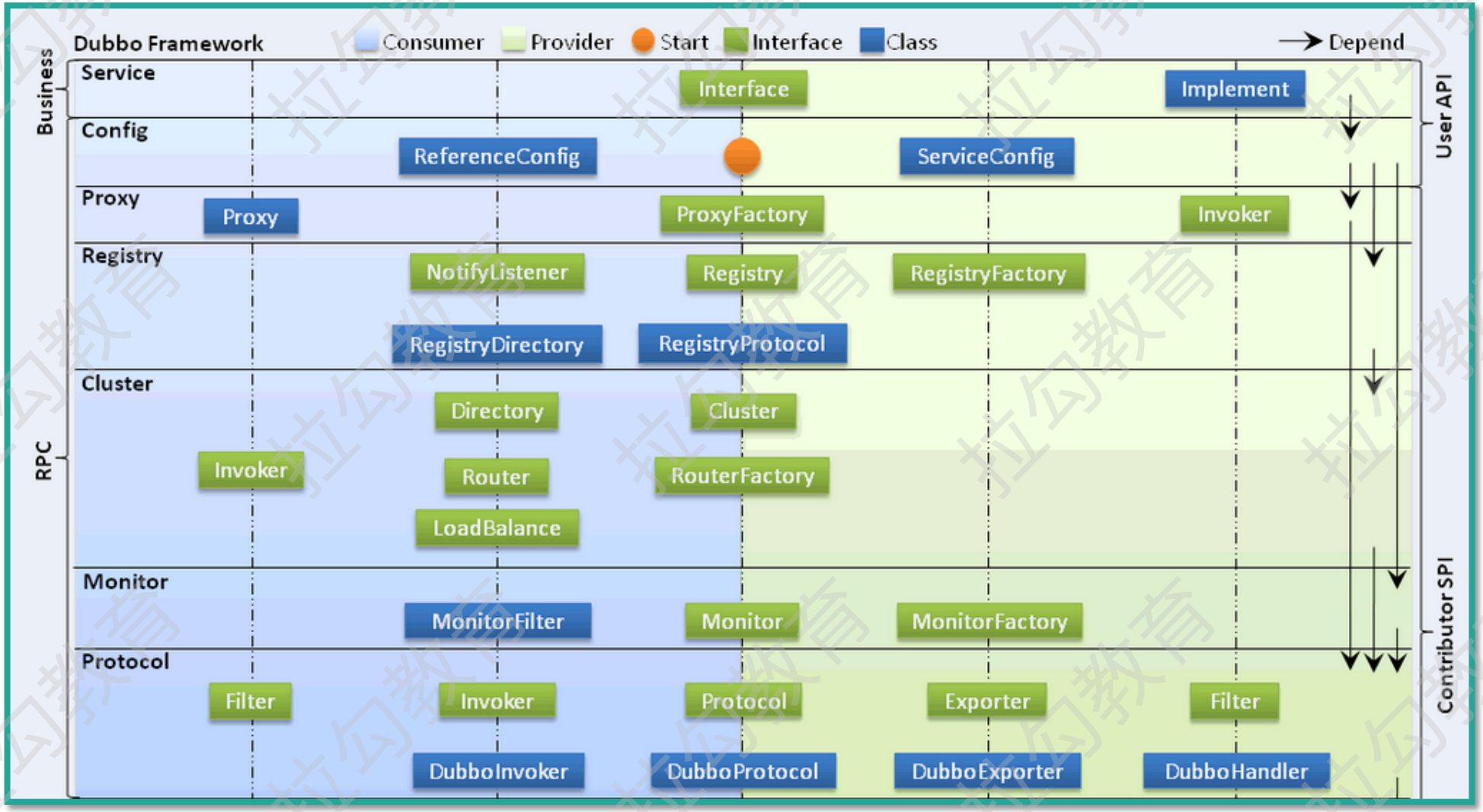
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



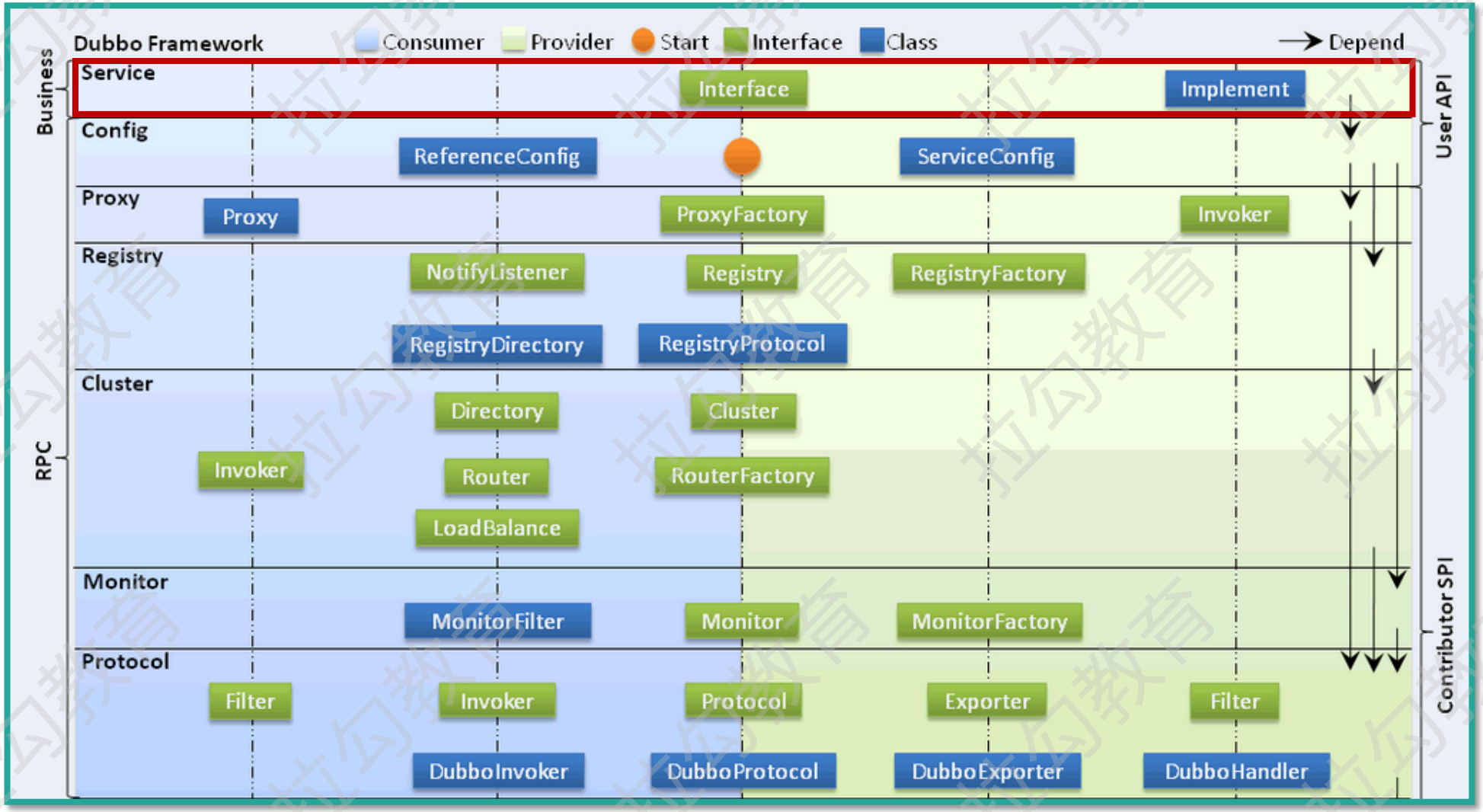
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



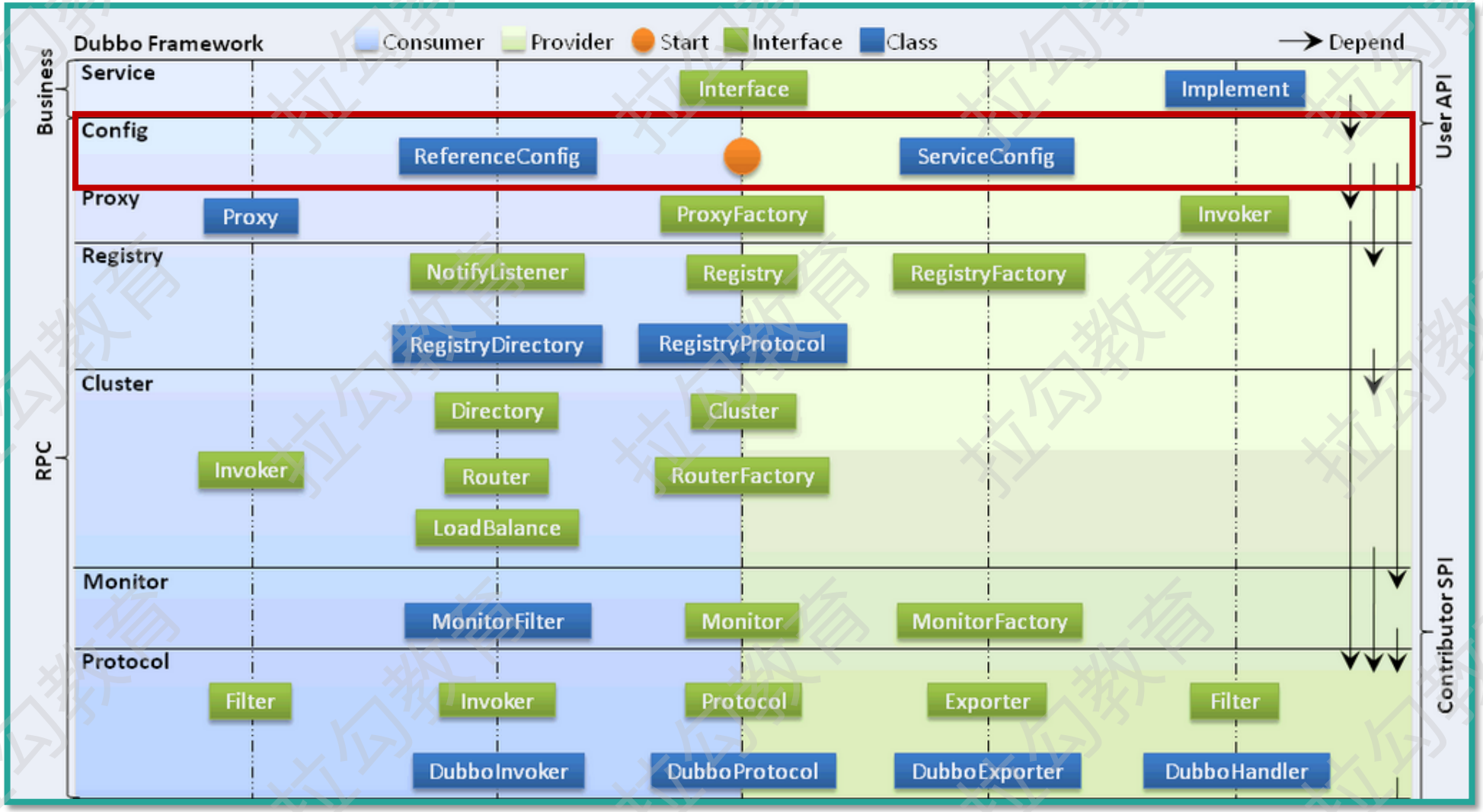
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



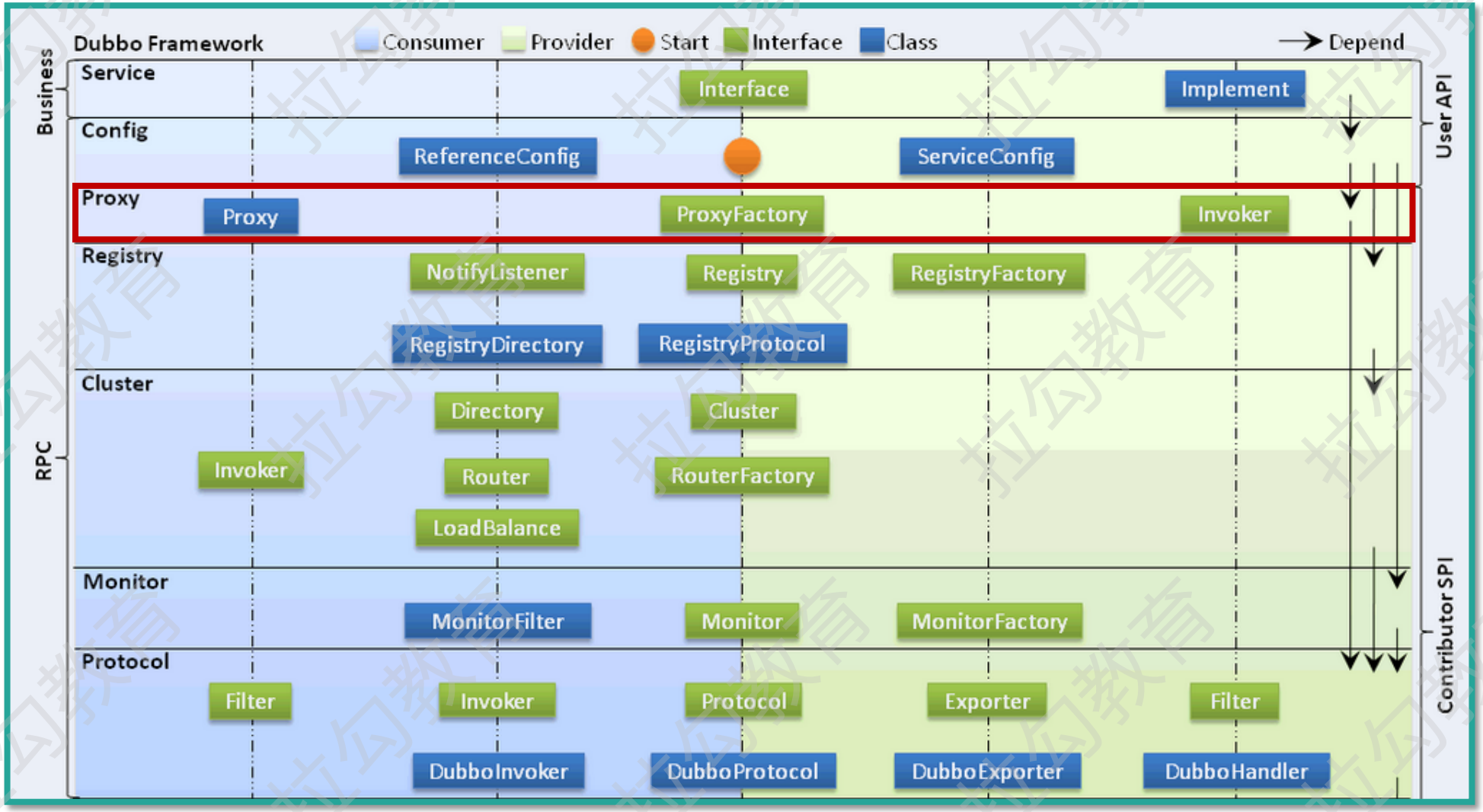
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



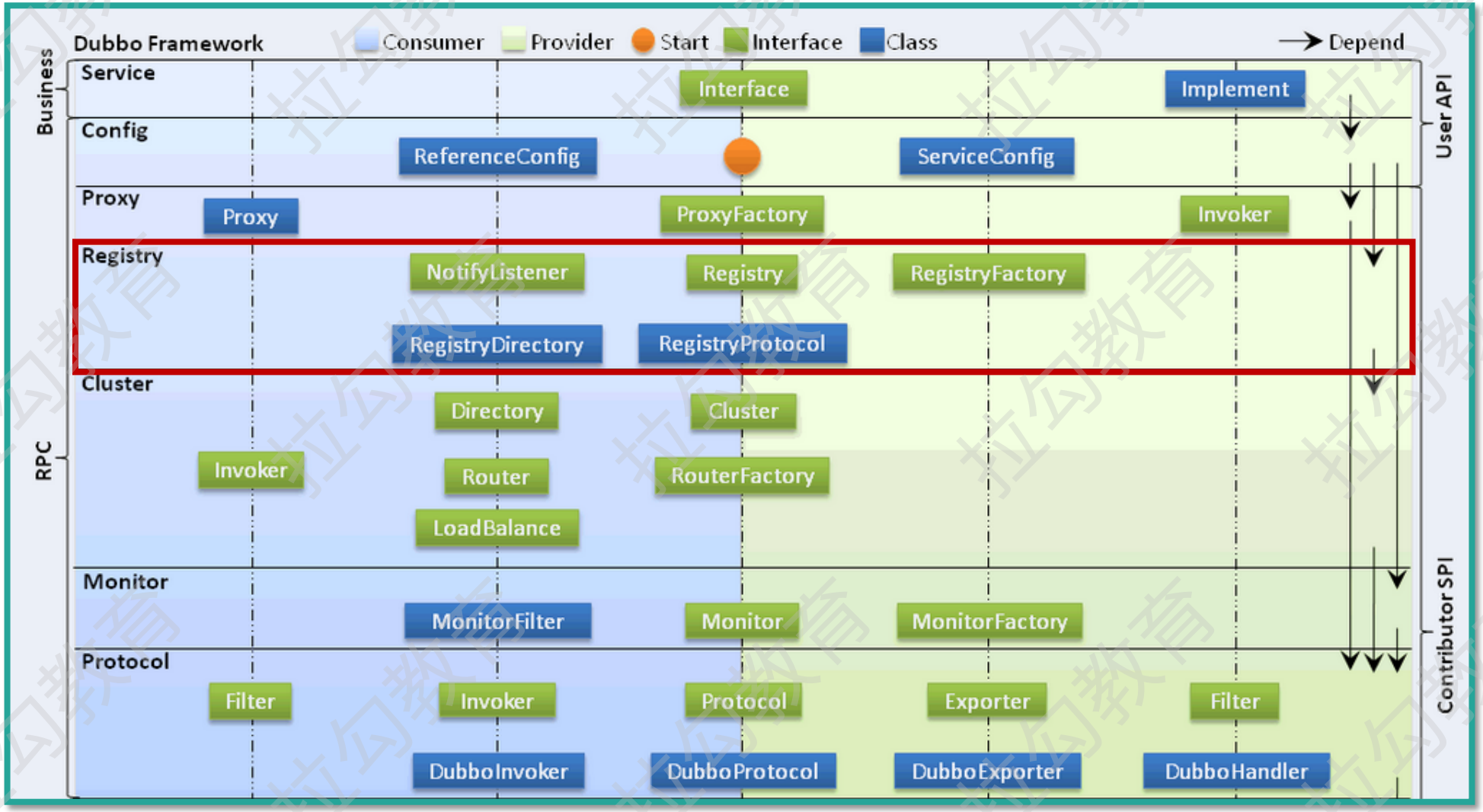
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



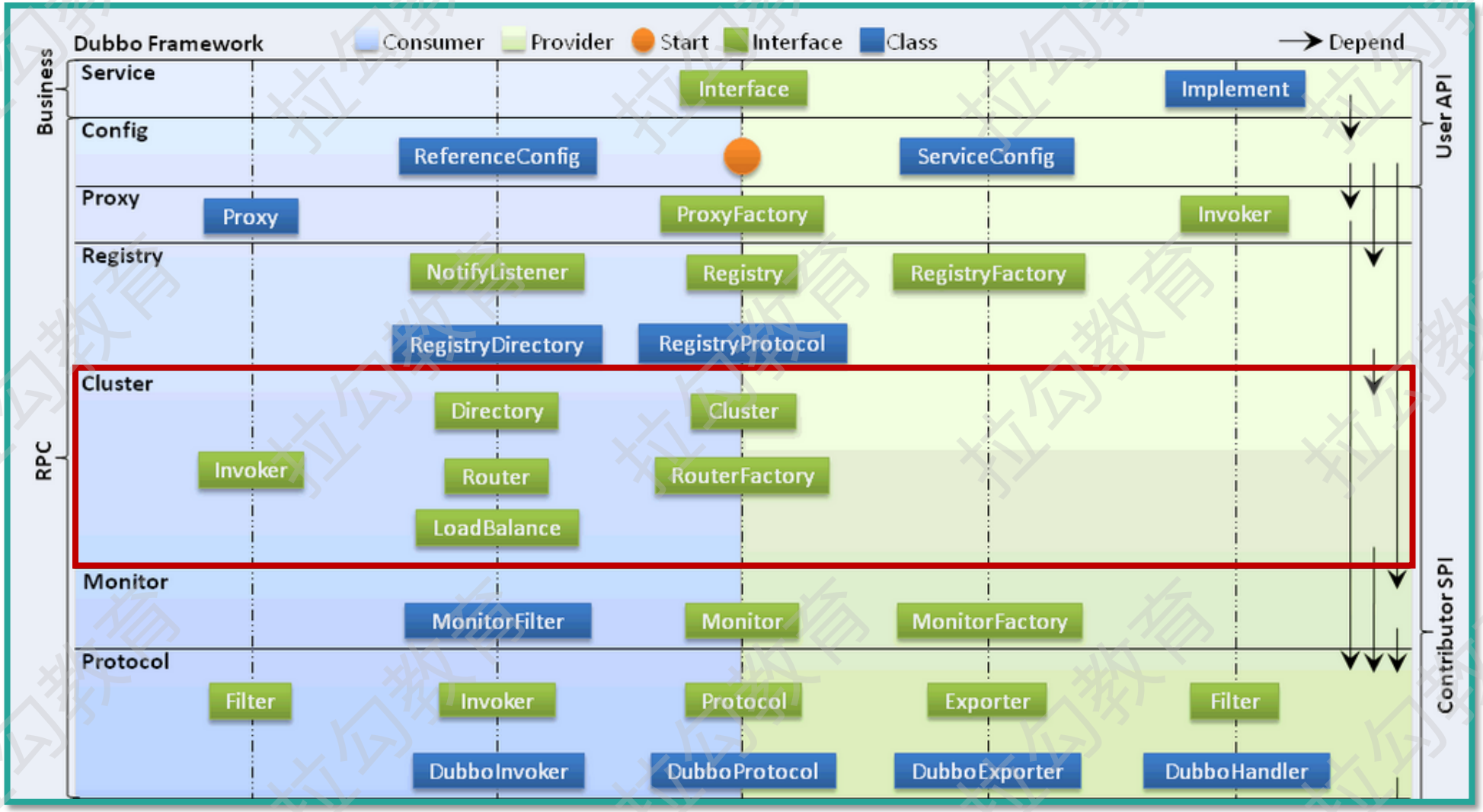
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



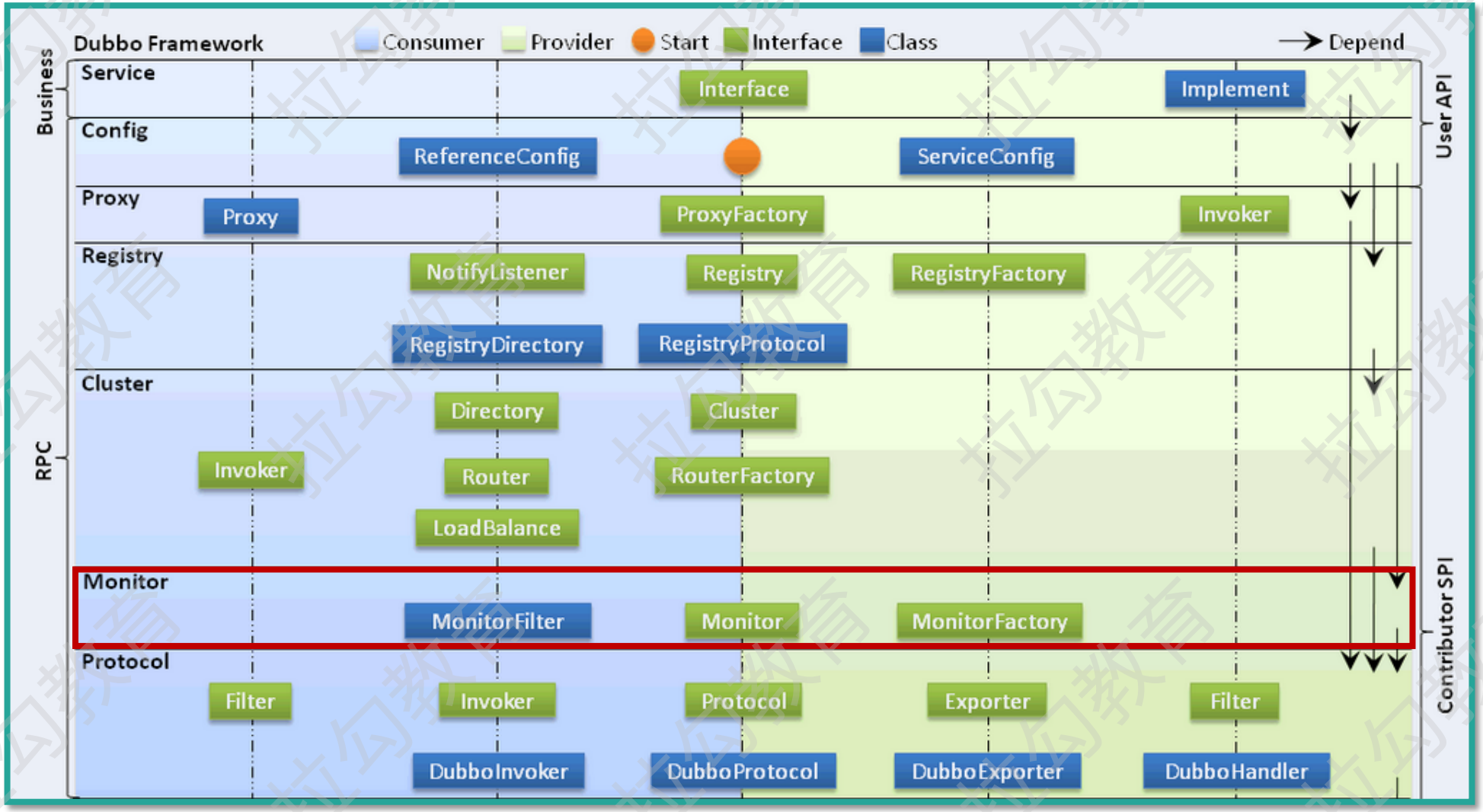
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



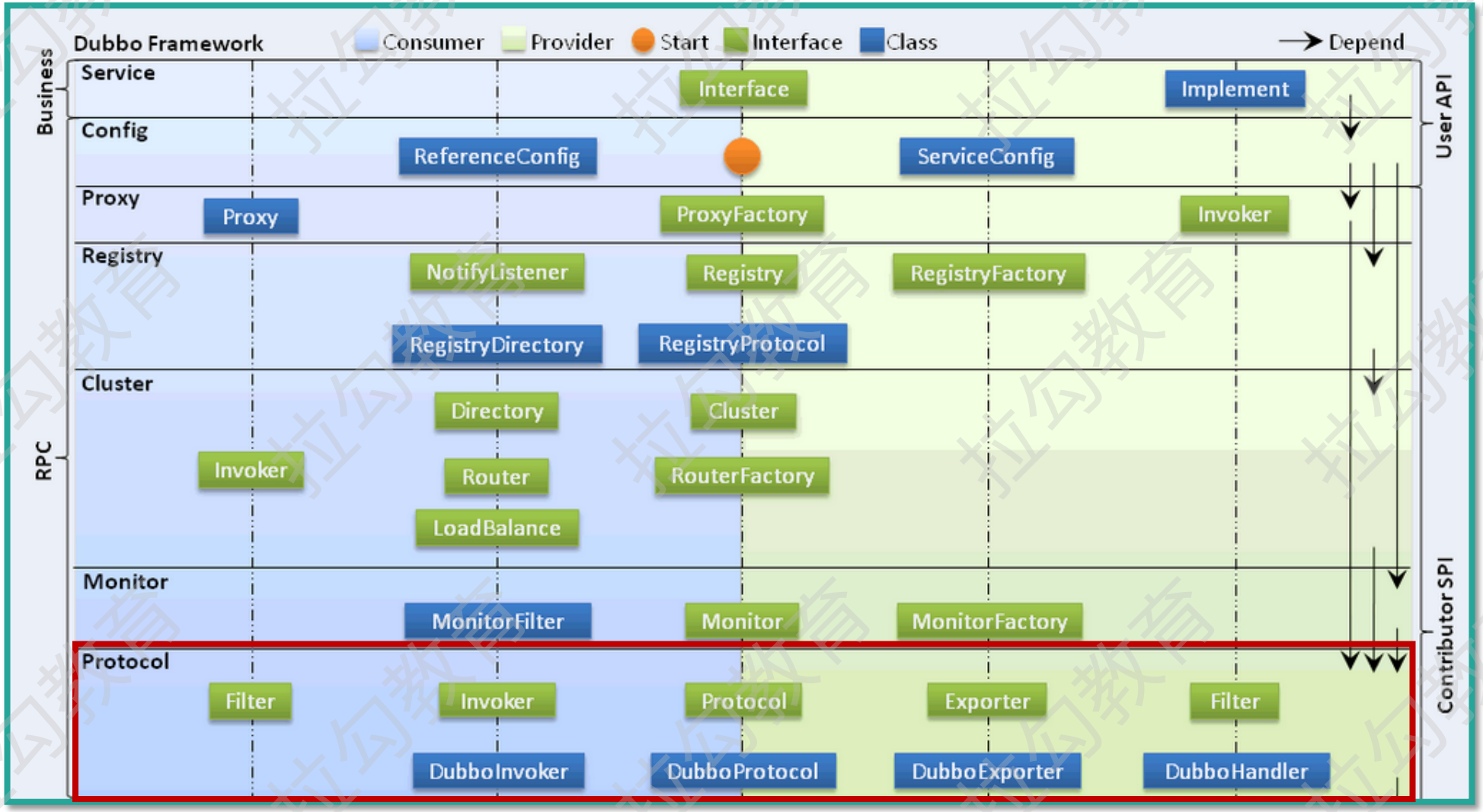
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



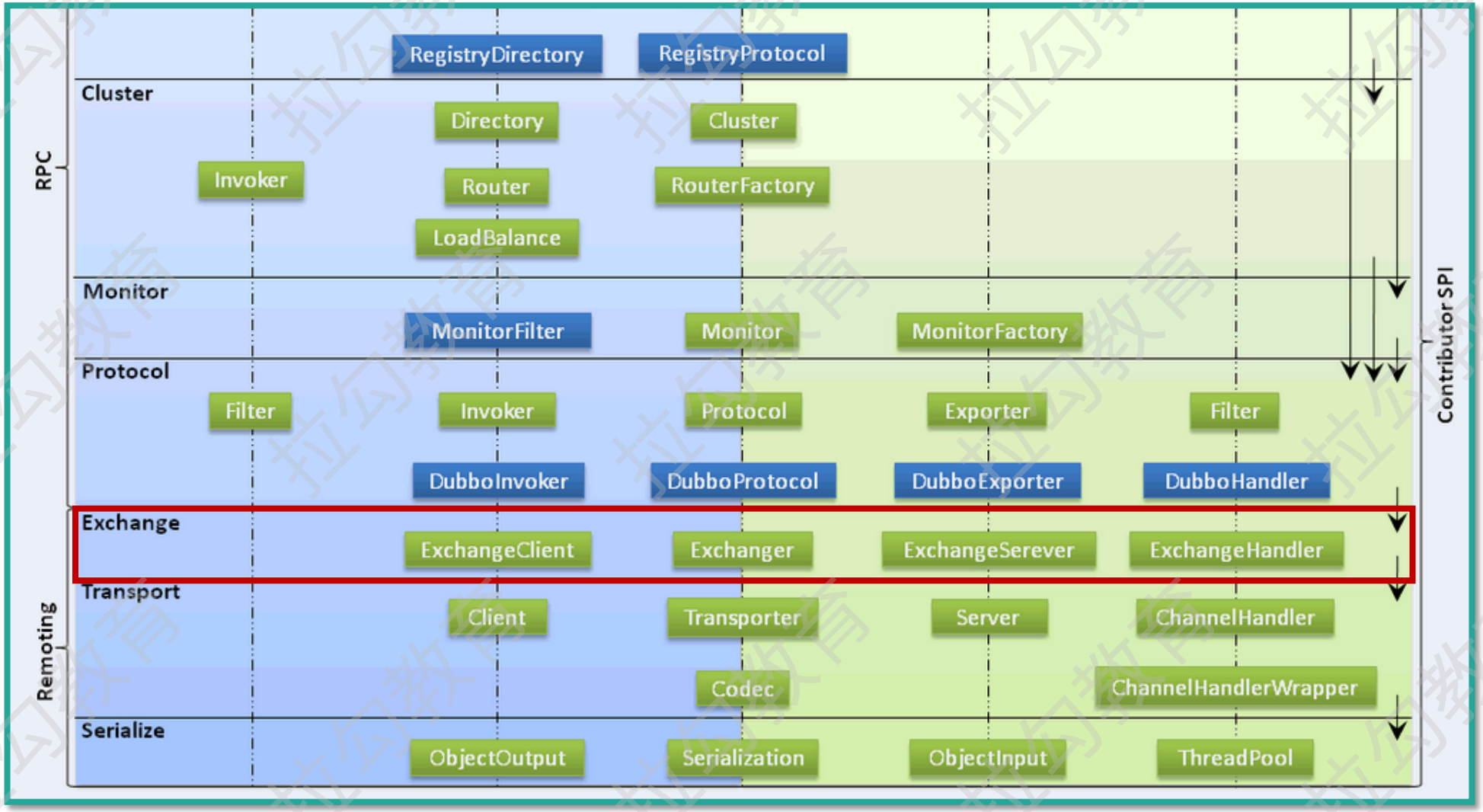
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



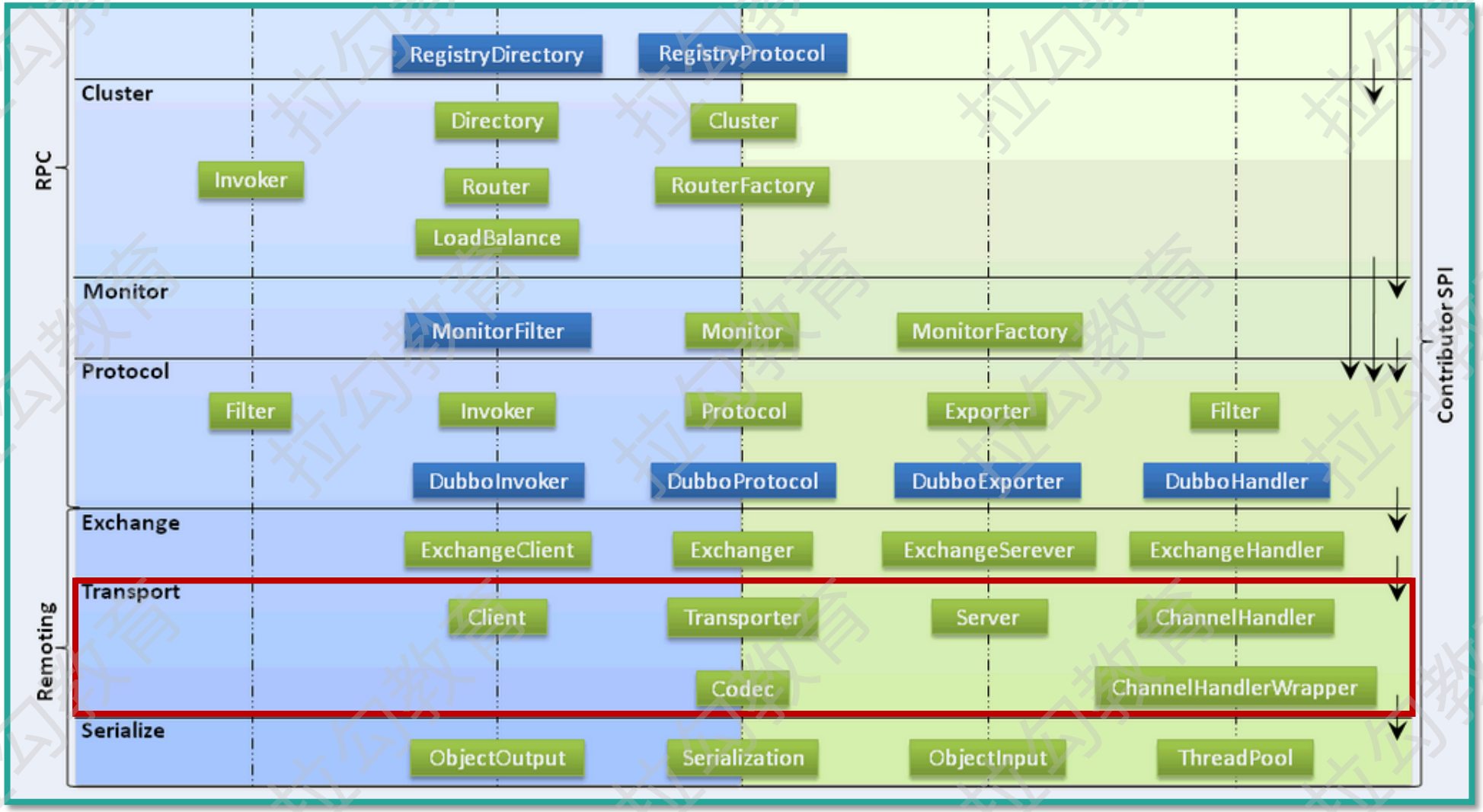
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



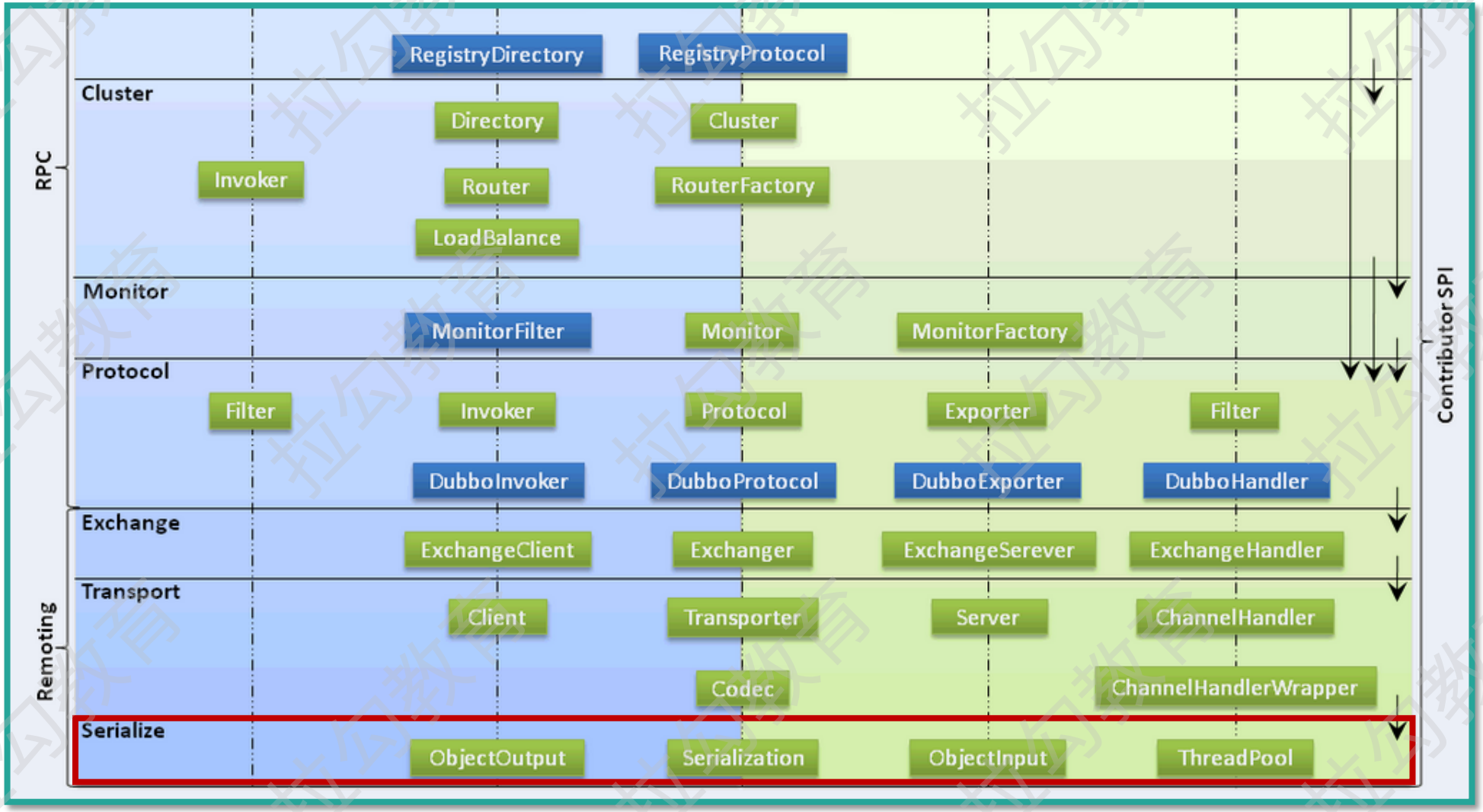
Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —

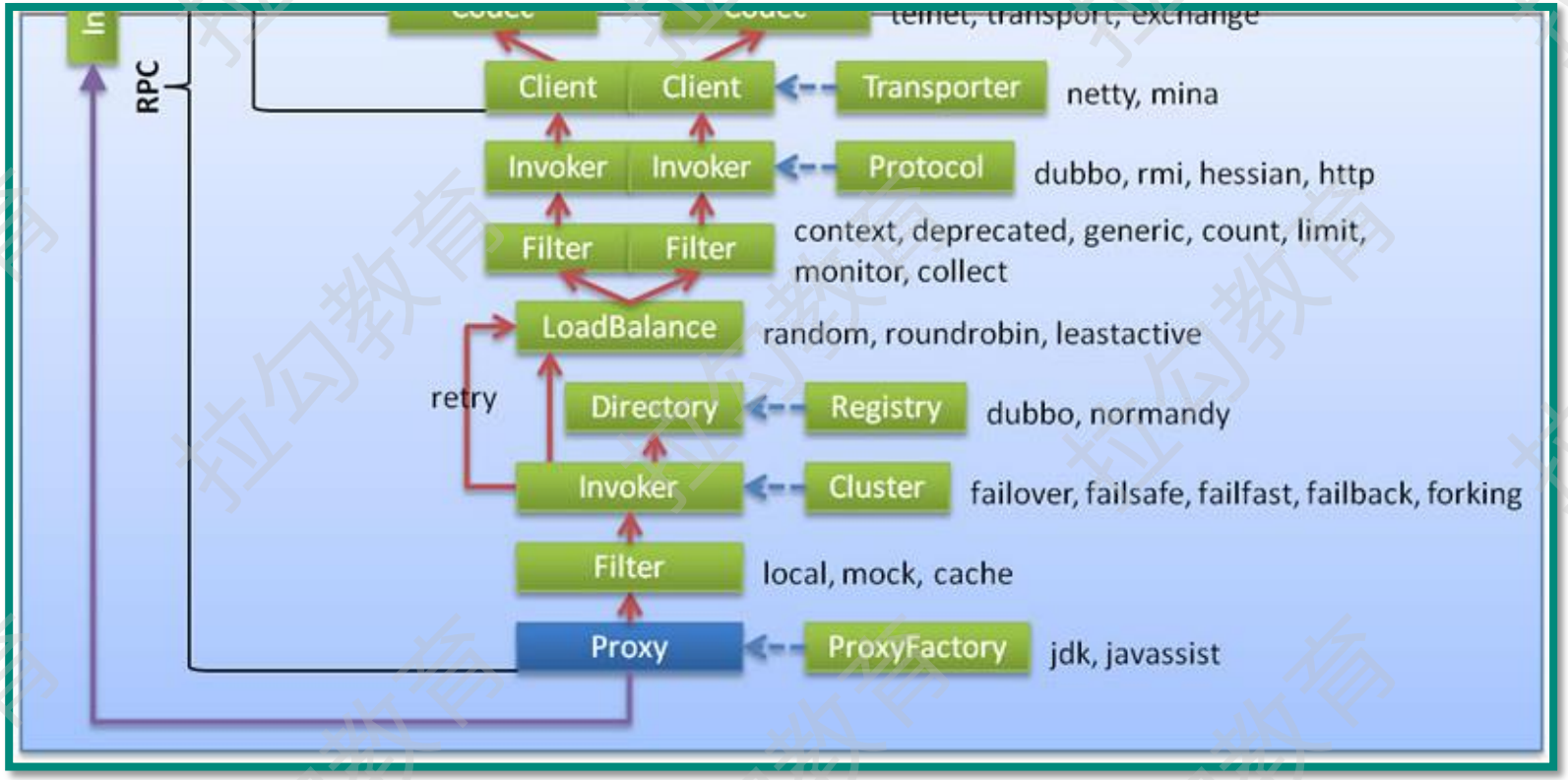


Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —

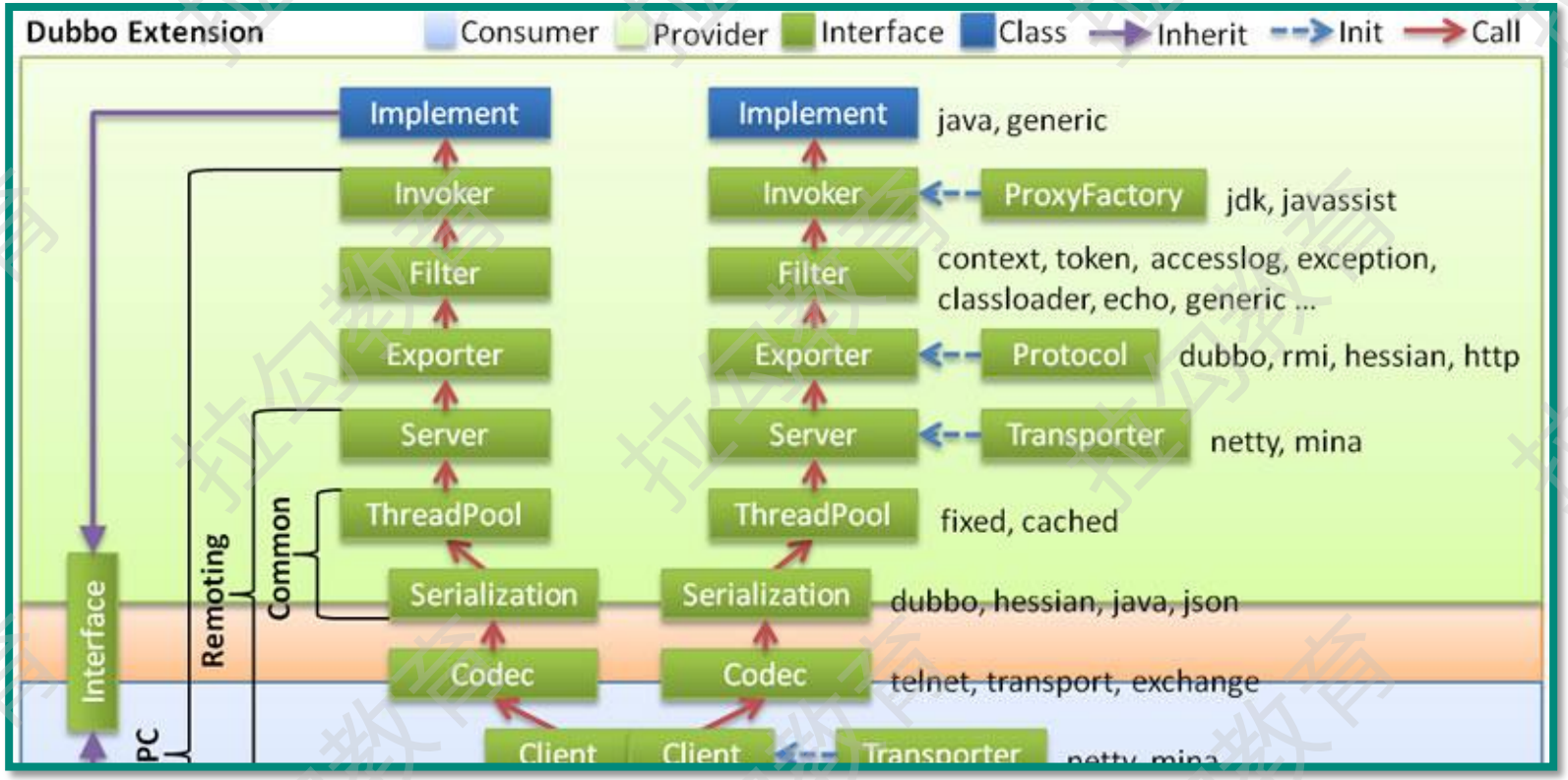


Dubbo 架构剖析


RPC
Client
Client
Transporter
netty, mina
Invoker
Invoker
Protocol
dubbo, rmi, hessian, http
Filter
Filter
context, deprecated, generic, count, limit, monitor, collect
LoadBalance
random, roundrobin, leastactive
retry
Directory
Registry
dubbo, normandy
Invoker
Cluster
failover, failsafe, failfast, failback, forking
Filter
local, mock, cache
Proxy
ProxyFactory
jdk, javassist

Dubbo 架构剖析

拉勾教育
— 互联网人实战大学 —



Dubbo Filter

拉勾教育
— 互联网人实战大学 —

Dubbo **官方针对 Filter** 做了很多的原生支持

常见的有：

- 打印访问日志（AccessLogFilter）
- 限流（ActiveLimitFilter、ExecuteLimitFilter、TpsLimitFilter）
- 监控功能（MonitorFilter）
- 异常处理（ExceptionFilter）

Dubbo Filter

拉勾教育
— 互联网人实战大学 —

```
private static <T> Invoker<T> buildInvokerChain(final Invoker<T>
    invoker, String key, String group) {
  Invoker<T> last = invoker; //最开始的last是指向invoker参数
  // 通过SPI方式加载Filter
  List<Filter> filters = ExtensionLoader
      .getExtensionLoader(Filter.class)
       .getActivateExtension(invoker.getUrl(), key, group);
  //遍历filters集合，将Filter封装成Invoker并串联成一个Filter链表
  for (int i = filters.size() - 1; i >= 0; i--) {
    final Filter filter = filters.get(i);
    final Invoker<T> next = last;
    last = new Invoker<T>() {
      @Override
      public Result invoke(Invocation invocation) {
        // 执行当前Filter的逻辑，在Filter中会调用下一个
        // Invoker.invoke()方法，触发下一个 Filter
        return filter.invoke(next, invocation);
      }
      //其他方法的实现都委托给了invoker参数(略)
    };
  }
  return last;
}
```

Dubbo Filter

拉勾教育
— 互联网人实战大学 —

- **Method**
  - buildInvokerChain(Invoker<T>, String, String)
- **Found usages** 2 usages
  - ProtocolFilterWrapper 2 usages
    - export(Invoker<T>) 1 usage

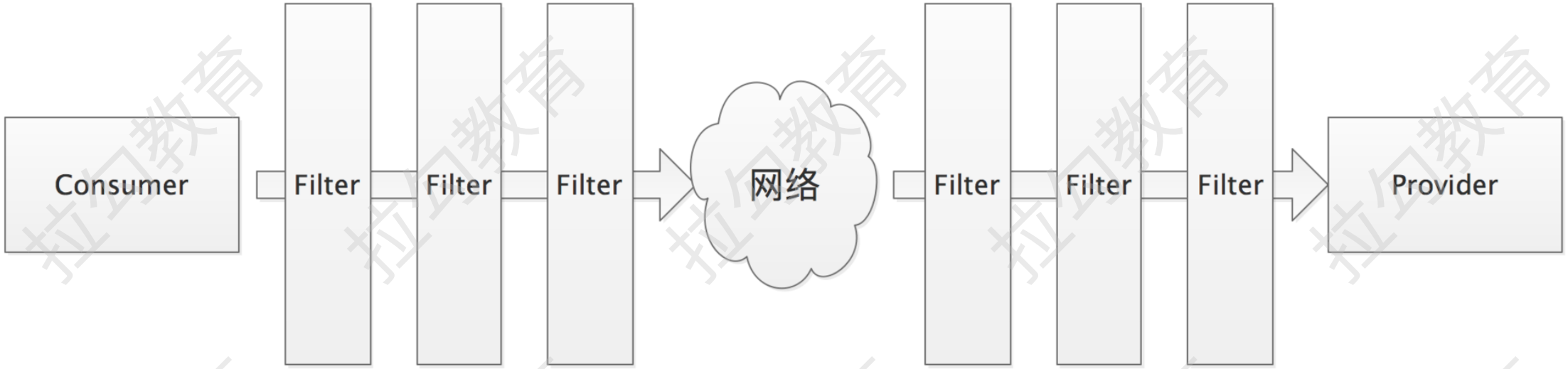
```
100 return protocol.export(buildInvokerChain(invoker, Constants.SERVICE_FILTER_KEY, Constants.PROVIDER));
```

    - refer(Class<T>, URL) 1 usage

```
108 return buildInvokerChain(protocol.refer(type, url), Constants.REFERENCE_FILTER_KEY, Constants.CONSUMER);
```

Dubbo Filter

拉勾教育
— 互联网人实战大学 —



Dubbo Filter

拉勾教育
— 互联网人实战大学 —

```
public Result invoke(Invoker<?> invoker, Invocation invocation) {
    RpcContext context = RpcContext.getContext();
    String remoteHost = context.getRemoteHost();
    long start = System.currentTimeMillis(); //记录请求的起始时间
    getConcurrent(invoker, invocation).incrementAndGet();//增加当前并发数
    try {
        Result result = invoker.invoke(invocation); // ?è????Filter
        // 收集监控信息
        collect(invoker, invocation, result, remoteHost,
            start, false);
        return result;
    } catch (RpcException e) {
        collect(invoker, invocation, null, remoteHost, start, true);
        throw e;
    } finally { // 减少当前并发数
        getConcurrent(invoker, invocation).decrementAndGet();
    }
}
```

Dubbo Filter

拉勾教育
— 互联网人实战大学 —

```
private void collect(Invoker<?> invoker, Invocation invocation,
    Result result, String remoteHost, long start, boolean error) {
  URL monitorUrl = invoker.getUrl()
    .getUrlParameter(Constants.MONITOR_KEY);
  Monitor monitor = monitorFactory.getMonitor(monitorUrl);
  //将请求的耗时时长、当前并发线程数以及请求结果等信息拼接到URL中
  URL statisticsURL = createStatisticsUrl(invoker, invocation,
    result, remoteHost, start, error);
  monitor.collect(statisticsURL); // 在DubboMonitor中缓存该URL
}
```

SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —

- 如果处于 Consumer 端，则会将当前 TracingContext 上下文序列化成 ContextCarrier 字符串

  并填充到 RpcContext 中

  RpcContext 中携带的信息会在之后随 Dubbo 请求一起发送出去，相应的，还会创建 ExitSpan

- 如果处于 Provider 端，则会从请求中反序列化 ContextCarrier 字符串

  并填充当前 TracingContext 上下文

  相应的，创建 EntrySpan

SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —

```
public void beforeMethod(EnhancedInstance objInst, Method method,
    Object[] allArguments, Class<?>[] argumentsTypes,
      MethodInterceptResult result) throws Throwable {
  Invoker invoker = (Invoker)allArguments[0]; // invoke()方法的两个参数
  Invocation invocation = (Invocation)allArguments[1];
  // RpcConterxt是Dubbo用来记录请求上下文信息的对象
  RpcContext rpcContext = RpcContext.getContext();
  //检测当前服务是Consumer端还是Provider端
  boolean isConsumer = rpcContext.isConsumerSide();
  URL requestURL = invoker.getUrl();
  AbstractSpan span;
  final String host = requestURL.getHost();
  final int port = requestURL.getPort();
  if (isConsumer) { // 检测是否 Consumer
    final ContextCarrier contextCarrier = new ContextCarrier();
    //如果当前是Consumer侧，则需要创建ExitSpan对象，其中EndpointName是
    //由请求URL地址、服务名以及方法名拼接而成的
    span = ContextManager.createExitSpan(
      generateOperationName(requestURL, invocation),
        contextCarrier, host + ":" + port);
```

SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —

```
    //创建CarrierItem链表，其中会根据当前Agent支持的版本号对
    // ContextCarrier进行序列化，该过程在前文已经详细介绍过了
    CarrierItem next = contextCarrier.items();
    while (next.hasNext()) {
        next = next.next();
        //将ContextCarrier字符串填充到RpcContext中，后续会随Dubbo请求一
        // 起发出
        rpcContext.getAttachments().put(next.getHeadKey(),
            next.getHeadValue());
    }
} else { //如果当前是Provider侧，则尝试从
    ContextCarrier contextCarrier = new ContextCarrier();
    CarrierItem next = contextCarrier.items();//创建CarrierItem链表
    while (next.hasNext()) {
        next = next.next();
        //从RpcContext中获取ContextCarrier字符串反序列化，并填充当前上
        //面创建的空白ContextCarrier对象
        next.setHeadValue(rpcContext
            .getAttachment(next.getHeadKey()));
    }
```

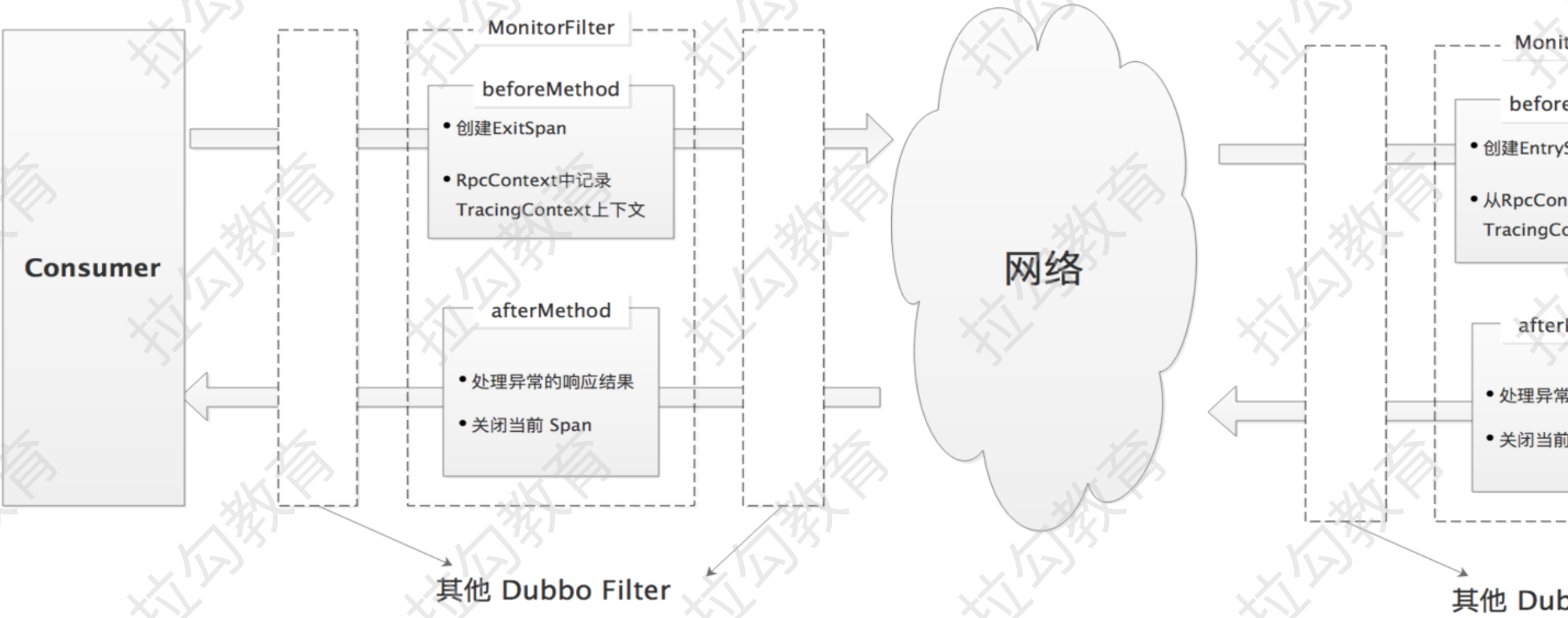
SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —

```
    }
  } else { //如果当前是Provider侧，则尝试从
    ContextCarrier contextCarrier = new ContextCarrier();
    CarrierItem next = contextCarrier.items();//创建CarrierItem链表
    while (next.hasNext()) {
      next = next.next();
      //从RpcContext中获取ContextCarrier字符串反序列化，并填充当前上
      //面创建的空白ContextCarrier对象
      next.setHeadValue(rpcContext
          .getAttachment(next.getHeadKey()));
    }
    //创建 EntrySpan，这个过程在前面分析Tomcat插件的时候，详细分析过了
    span = ContextManager.createEntrySpan(generateOperationName(
      requestURL, invocation), contextCarrier);
  }
  // 设置 Tags
  Tags.URL.set(span, generateRequestURL(requestURL, invocation));
  span.setComponent(ComponentsDefine.DUBBO);//设置 component
  SpanLayer.asRPCFramework(span); //设置 SpanLayer
}
```

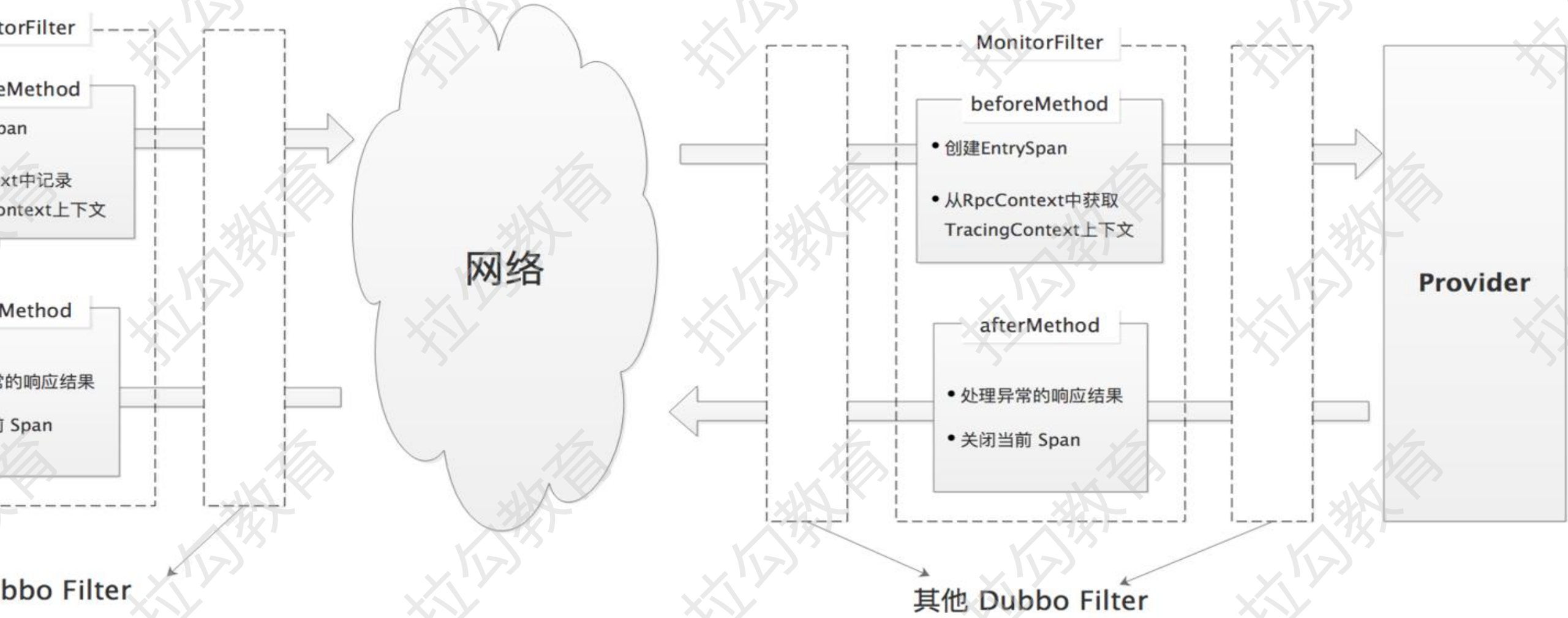
SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —



SkyWalking Dubbo 插件

拉勾教育
— 互联网人实战大学 —



Next：第16讲《带你揭开 toolkit-activation 工具箱的秘密》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U