首先 beats 的所有 output 都在libbeat中被实现，libbeat 属于公共包，其他 beats 的 output 依赖于这个公共包,不同的Beats会根据相关配置在 output group 中找到已有的 output 实例。

不同的output 插件通过实现公共的类型和接口来实现不同的output实例，再通过注册的方式使不同的output插件可用。需要实现的公共接口在outputs.go文件中，如下：

```
type Client interface {
  // 关闭client
  Close() error

  // 此方法发送事件给 client sink, 一旦所有的事件被处理完成，client 必须同步或异步
  // 对给出的 batch 进行 ACK 确认。client 使用 Retry/Cancelled 获取一批 publisher
  // pipeline 中未处理的事件。publisher pipeline（如果在 output factory 配置了）
  // 会处理好 retrying/dropping 事件。
  Publish(publisher.Batch) error

  // 识别 client 类型和 endpoint
  String() string
}

// NetworkClient defines the required client capabilities for network based
// outputs, that must be reconnectable.
type NetworkClient interface {
  Client
  Connectable
}

// Connectable is optionally implemented by clients that might be able to close
// and reconnect dynamically.
type Connectable interface {
  // Connect establishes a connection to the clients sink.
  // The connection attempt shall report an error if no connection could been
  // established within the given time interval. A timeout value of 0 == wait
  // forever.
```

```
29  Connect() error
30 }
```

解析 kafka broken的开发流程如下：

client 接口实现

kafka broken 在 client.go 中实现了 outputs.go 定义的接口以及自定义的接口，如下所示：

```
1 func newKafkaClient // 构造kafka client
2 func Connect // 异步连接Kafka server
3 func Close // 关闭client
4 func Publish // 发布数据
5 func String // kafka hosts 信息
6 func getEventMessage // 获取事件消息->给 Publish 使用
7 func successWorker // 处理消息发送成功后 ACK 任务
8 func errorWorker // producer 处理任务失败消息
```

服务注册

初始化

```
1  # libbeat/outputs/kafka/kafka.go:54
2  # 注册服务名为 kafka, 指定处理方法为 makeKafka
3  outputs.RegisterType("kafka", makeKafka)
4
5  ## libbeat/publisher/includes/includes.go:18
6  # 添加到includes中
7  import (
8   // import queue types
9   _ "github.com/elastic/beats/libbeat/outputs/codec/format"
10   _ "github.com/elastic/beats/libbeat/outputs/codec/json"
11   _ "github.com/elastic/beats/libbeat/outputs/console"
12   _ "github.com/elastic/beats/libbeat/outputs/elasticsearch"
13   _ "github.com/elastic/beats/libbeat/outputs/fileout"
14   _ "github.com/elastic/beats/libbeat/outputs/kafka"
15   _ "github.com/elastic/beats/libbeat/outputs/kafkaRest"
16   _ "github.com/elastic/beats/libbeat/outputs/logstash"
17   _ "github.com/elastic/beats/libbeat/outputs/redis"
```

```
 _ "github.com/elastic/beats/libbeat/publisher/queue/memqueue"
 _ "github.com/elastic/beats/libbeat/publisher/queue/spool"
)
```

makeKafka

1. 指定参数与返回值

```
# libbeat/outputs/kafka/kafka.go:72
func makeKafka(
 _ outputs.IndexManager, // IndexManager provides additional index related services to the outputs.
 beat beat.Info, // Info stores a beats instance meta data.
 observer outputs.Observer,// Observer provides an interface used by outputs to report common events on
 // documents/events being published and I/O workload.
 cfg *common.Config, // Config object to store hierarchical configurations into.
) (outputs.Group, error) // Group configures and combines multiple clients into load-balanced group of clients
 // being managed by the publisher pipeline.
```

2. 读取并解析配置文件

```
# 读取配置文件返回 kafkaconfig (libbeat/outputs/kafka/config.go:39)
config, err := readConfig(cfg)
if err != nil {
 return outputs.Fail(err)
}
#
topic, err := outil.BuildSelectorFromConfig(cfg, outil.Settings{
 Key: "topic",
 MultiKey: "topics",
 EnableSingleOnly: true,
 FailEmpty: true,
})
if err != nil {
 return outputs.Fail(err)
}

libCfg, err := newSaramaConfig(config)
if err != nil {
 return outputs.Fail(err)
```

```
}

hosts, err := outputs.ReadHostList(cfg)
if err != nil {
 return outputs.Fail(err)
}

codec, err := codec.CreateEncoder(beat, config.Codec)
if err != nil {
 return outputs.Fail(err)
}
```

output 插件初始化

```
libbeat/outputs/plugin.go:38
func init() {}
```