一、alertmanager配置文件介绍

- 网上案例：https://www.jianshu.com/p/6b05104278cc

```
# global块配置下的配置选项在本配置文件内的所有配置项下可见
global:
  # 在Alertmanager内管理的每一条告警均有两种状态: "resolved"或者"firing". 在altermanager首次发送告警通知后, 该告警会一直处于firing状态,设置resolve_timeout可以指定处于firing状态的告警间隔多长时间会被设置为resolved状态, 在设置为resolved状态的告警后,altermanager不会再发送firing的告警通知.
  resolve_timeout: 1h

  # 邮件告警配置
  smtp_smarthost: 'smtp.exmail.qq.com:25'
  smtp_from: 'dukuan@xxx.com'
  smtp_auth_username: 'dukuan@xxx.com'
  smtp_auth_password: 'DKxxx'
  # HipChat告警配置
  # hipchat_auth_token: '123456789'
  # hipchat_auth_url: 'https://hipchat.foobar.org/'
  # wechat
  wechat_api_url: 'https://qyapi.weixin.qq.com/cgi-bin/'
  wechat_api_secret: 'JJ'
  wechat_api_corp_id: 'ww'

  # 告警通知模板
templates:
- '/etc/alertmanager/config/*.tmpl'

# route: 根路由,该模块用于该根路由下的节点及子路由routes的定义. 子树节点如果不对相关配置进行配置，则默认会从父路由树继承该配置选项。每一条告警都要进入route，即要求配置选项group_by的值能够匹配到每一条告警的至少一个labelkey(即通过POST请求向altermanager服务接口所发送告警的labels项所携带的<labelname>)，告警进入到route后，将会根据子路由routes节点中的配置项match_re或者match来确定能进入该子路由节点的告警(由在match_re或者match下配置的labelkey: labelvalue是否为告警labels的子集决定，是的话则会进入该子路由节点，否则不能接收进入该子路由节点).
route:
  # 例如所有labelkey:labelvalue含cluster=A及altertname=LatencyHigh labelkey的告警都会被归入单一组中
  group_by: ['job', 'altername', 'cluster', 'service','severity']
  # 若一组新的告警产生，则会等group_wait后再发送通知，该功能主要用于当告警在很短时间内接连产生时，在group_wait内合并为单一的告警后再发送
  group_wait: 30s
  # 再次告警时间间隔
  group_interval: 5m
  # 如果一条告警通知已成功发送，且在间隔repeat_interval后，该告警仍然未被设置为resolved，则会再次发送该告警通知
  repeat_interval: 12h
  # 默认告警通知接收者，凡未被匹配进入各子路由节点的告警均被发送到此接收者
  receiver: 'wechat'
  # 上述route的配置会被传递给子路由节点，子路由节点进行重新配置才会被覆盖

  # 子路由树
  routes:
```

```
  # 该配置选项使用正则表达式来匹配告警的labels，以确定能否进入该子路由树
  # match_re和match均用于匹配labelkey为service,labelvalue分别为指定值的告警，被匹配到的告警会将通知发送到对应的receiver
  - match_re:
      service: ^(foo1|foo2|baz)$
    receiver: 'wechat'
    # 在带有service标签的告警同时有severity标签时，他可以有自己的子路由，同时具有severity != critical的告警则被发送给接收者team-ops-mails,对severity == critical的告警则被发送到对应的接收者即team-ops-pager
    routes:
    - match:
        severity: critical
      receiver: 'wechat'
  # 比如关于数据库服务的告警，如果子路由没有匹配到相应的owner标签，则都默认由team-DB-pager接收
  - match:
      service: database
    receiver: 'wechat'
  # 我们也可以先根据标签service:database将数据库服务告警过滤出来，然后进一步将所有同时带labelkey为database
  - match:
      severity: critical
    receiver: 'wechat'
# 抑制规则，当出现critical告警时 忽略warning
inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'warning'
  # Apply inhibition if the alertname is the same.
  #   equal: ['alertname', 'cluster', 'service']
  #
# 收件人配置
receivers:
- name: 'team-ops-mails'
  email_configs:
  - to: 'dukuan@xxx.com'
- name: 'wechat'
  wechat_configs:
  - send_resolved: true
    corp_id: 'ww'
    api_secret: 'JJ'
    to_tag: '1'
    agent_id: '1000002'
    api_url: 'https://qyapi.weixin.qq.com/cgi-bin/'
    message: '{{ template "wechat.default.message" . }}'
#- name: 'team-X-pager'
#  email_configs:
#  - to: 'team-X+alerts-critical@example.org'
#  pagerduty_configs:
#  - service_key: <team-X-key>
#
#- name: 'team-Y-mails'
#  email_configs:
#  - to: 'team-Y+alerts@example.org'
#
#- name: 'team-Y-pager'
#  pagerduty_configs:
```

```
#  - service_key: <team-Y-key>
#
#- name: 'team-DB-pager'
#  pagerduty_configs:
#  - service_key: <team-DB-key>
#
#- name: 'team-X-hipchat'
#  hipchat_configs:
#  - auth_token: <auth_token>
#    room_id: 85
#    message_format: html
#    notify: true
```

二、alertmanager route配置详解

```
# route: 根路由,该模块用于该根路由下的节点及子路由routes的定义. 子树节点如果不对相关配置进行配置，则默认会从父路由树继承该配置选项。每一条告警都要进入route，即要求配置选项group_by的值能够匹配到每一条告警的至少一个labelkey(即通过POST请求向altermanager服务接口所发送告警的labels项所携带的<labelname>)，告警进入到route后，将会根据子路由routes节点中的配置项match_re或者match来确定能进入该子路由节点的告警(由在match_re或者match下配置的labelkey: labelvalue是否为告警labels的子集决定，是的话则会进入该子路由节点，否则不能接收进入该子路由节点).
route:
  # 例如所有labelkey:label value含cluster=A及altertname=LatencyHigh labelkey的告警都会被归入单一组中
  group_by: ['job', 'altername', 'cluster', 'service','severity']
  # 若一组新的告警产生，则会等group_wait后再发送通知，该功能主要用于当告警在很短时间内接连产生时，在group_wait内合并为单一的告警后再发送
  group_wait: 30s
  # 再次告警时间间隔
  group_interval: 5m
  # 如果一条告警通知已成功发送，且在间隔repeat_interval后，该告警仍然未被设置为resolved，则会再次发送该告警通知
  repeat_interval: 12h
  # 默认告警通知接收者，凡未被匹配进入各子路由节点的告警均被发送到此接收者
  receiver: 'wechat'
  # 上述route的配置会被传递给子路由节点，子路由节点进行重新配置才会被覆盖

  # 子路由树
  routes:
  # 该配置选项使用正则表达式来匹配告警的labels，以确定能否进入该子路由树
  # match_re和match均用于匹配labelkey为service,labelvalue分别为指定值的告警，被匹配到的告警会将通知发送到对应的receiver
  - match_re:
      service: ^(foo1|foo2|baz)$
    receiver: 'wechat'
    # 在带有service标签的告警同时有severity标签时，他可以有自己的子路由，同时具有severity != critical的告警则被发送给接收者team-ops-mails,对severity == critical的告警则被发送到对应的接收者即team-ops-pager
    routes:
    - match:
        severity: critical
      receiver: 'wechat'
  # 比如关于数据库服务的告警，如果子路由没有匹配到相应的owner标签，则都默认由team-DB-pager接收
```

```
  - match:
      service: database
    receiver: 'wechat'
  # 我们也可以先根据标签service:database将数据库服务告警过滤出来，然后进一步将所有同时带
labelkey为database
  - match:
      severity: critical
    receiver: 'wechat'
# 抑制规则，当出现critical告警时 忽略warning
```

三、告警实战

3.1、邮件告警通知

- 以网易邮箱为例
  - 需要账号登录密码：xxxx
  - 需要账号SMTP密码：AMHOWTBONHLXWJRZ
- 编辑配置文件 【/root/kube-prometheus/manifests/alertmanager-secret.yaml】

```
[root@k8s-master01 manifests]# cat /root/kube-prometheus/manifests/alertmanager-
secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: alertmanager-main
  namespace: monitoring
stringData:
  email.tmpl: |-
    {{ define "email.html" }}
    {{- if gt (len .Alerts.Firing) 0 -}}
    {{- range $index, $alert := .Alerts -}}
    {{- if eq $index 0 -}}
    <h2>**********告警通知**********<h2>
    告警类型: {{ $alert.Labels.alertname }} <br>
    告警级别: {{ $alert.Labels.severity }} <br>
    {{- end }}
    ====================== <br>
    告警主题: {{ $alert.Annotations.summary }} <br>
    告警详情: {{ $alert.Annotations.description }} <br>
    故障时间: {{ ($alert.StartsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}
<br>
    {{ if gt (len $alert.Labels.instance) 0 -}}故障实例: {{
$alert.Labels.instance }}{{- end -}}
    {{- end }}
    {{- end }}
    {{- if gt (len .Alerts.Resolved) 0 -}}
    {{- range $index, $alert := .Alerts -}}
    {{- if eq $index 0 -}}
    <h2>**********恢复通知**********<h2>
    告警类型: {{ $alert.Labels.alertname }} <br>
    告警级别: {{ $alert.Labels.severity }} <br>
    {{- end }}
    ====================== <br>
    告警主题: {{ $alert.Annotations.summary }} <br>
```

```
    告警详情: {{ $alert.Annotations.description }} <br>
    故障时间: {{ ($alert.StartsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}
<br>
    恢复时间: {{ ($alert.EndsAt.Add 28800e9).Format "2006-01-02 15:04:05" }} <br>
    {{ if gt (len $alert.Labels.instance) 0 -}}故障实例: {{
$alert.Labels.instance }}{{- end -}}
    {{- end }}
    {{- end }}
    {{- end }}
  wechat.tmpl: |-
    {{ define "wechat.default.message" }}
    {{- if gt (len .Alerts.Firing) 0 -}}
    {{- range $index, $alert := .Alerts -}}
    ======== 异常告警 ========
    告警名称: {{ $alert.Labels.alertname }}
    告警级别: {{ $alert.Labels.severity }}
    告警机器: {{ $alert.Labels.instance }} {{ $alert.Labels.device }}
    告警详情: {{ $alert.Annotations.summary }}
    告警时间: {{ $alert.StartsAt.Format "2006-01-02 15:04:05" }}
    ========== END ==========
    {{- end }}
    {{- end }}
    {{- if gt (len .Alerts.Resolved) 0 -}}
    {{- range $index, $alert := .Alerts -}}
    ======== 告警恢复 ========
    告警名称: {{ $alert.Labels.alertname }}
    告警级别: {{ $alert.Labels.severity }}
    告警机器: {{ $alert.Labels.instance }}
    告警详情: {{ $alert.Annotations.summary }}
    告警时间: {{ $alert.StartsAt.Format "2006-01-02 15:04:05" }}
    恢复时间: {{ $alert.EndsAt.Format "2006-01-02 15:04:05" }}
    ========== END ==========
    {{- end }}
    {{- end }}
    {{- end }}
  alertmanager.yaml: |-
    "global":
      "resolve_timeout": "1m"
      # 添加邮件告警信息
      smtp_from: "tzh971204@163.com"
      smtp_smarthost: "smtp.163.com:465"
      smtp_hello: "163.com"
      smtp_auth_username: "tzh971204@163.com"
      smtp_auth_password: "AMHOWTBONHLXWJRZ"
      smtp_require_tls: false     # tls关闭
      # 添加企业微信告警信息
      wechat_api_url: "https://qyapi.weixin.qq.com/cgi-bin/"
      wechat_api_secret: "EhlNg2sDjCwAazKfs8_ozJ8Zj6uAcUezR6aVNMz18E0"
      wechat_api_corp_id: "wwf0e1c242baef1eaf"
    # 指定模板的路径
    templates:
    - "/etc/alertmanager/config/*.tmpl"
    "inhibit_rules":
    - "equal":
      - "namespace"
      - "alertname"
      "source_match":
        "severity": "critical"
```

```
      "target_match_re":
        "severity": "warning|info"
    - "equal":
      - "namespace"
      - "alertname"
      "source_match":
        "severity": "warning"
      "target_match_re":
        "severity": "info"
    "receivers":
    - name: "wechat"
      wechat_configs:
      - send_resolved: true  # 告警解决后是否发生个告警解决邮件
        agent_id: '1000002'  # AgentId
        to_tag: '2'          # 部门ID
        to_user: "@all"
        message: '{{ template "wechat.default.message" . }}'
    - "name": "Default"
    # 新增收件人信息
      "email_configs":             # 类型是邮件告警
      - to: "tzh971204@163.com"    # 收件邮件
        send_resolved: true        # 告警解决后是否发生个告警解决邮件
        html: '{{ template "email.html" . }}' # 模板
    - "name": "Watchdog"
    - "name": "Critical"
    "route":
      "group_by":
      - "namespace"
      # 新增2个告警分组规则
      - job
      - alertname
      "group_interval": "5m"    # 距离第一次发送告警，等待多久再次发送告警
      "group_wait": "30s"       # 距离第一次发送告警，等待多久再次发送告警
      "receiver": "Default"     # 告警接收人
      "repeat_interval": "30s" # 告警重发时间
      "routes":
      - "match":
          "alertname": "Watchdog"
        "receiver": "wechat"
      - "match":
          "severity": "critical"
        "receiver": "wechat"
type: Opaque
```

- replace即可，不需要重启po,然后可以去Alertmanager的ui查看配置是否加载成功【http://192.168.1.110:30585/#/status】
- 【http://192.168.1.110:30585/#/alerts】这查看是否分组成功
- 然后就可以去邮箱中查看告警邮件【若无邮件收到查看 kubectl logs -f -n monitoring alertmanager-main-0 -c alertmanager日志】

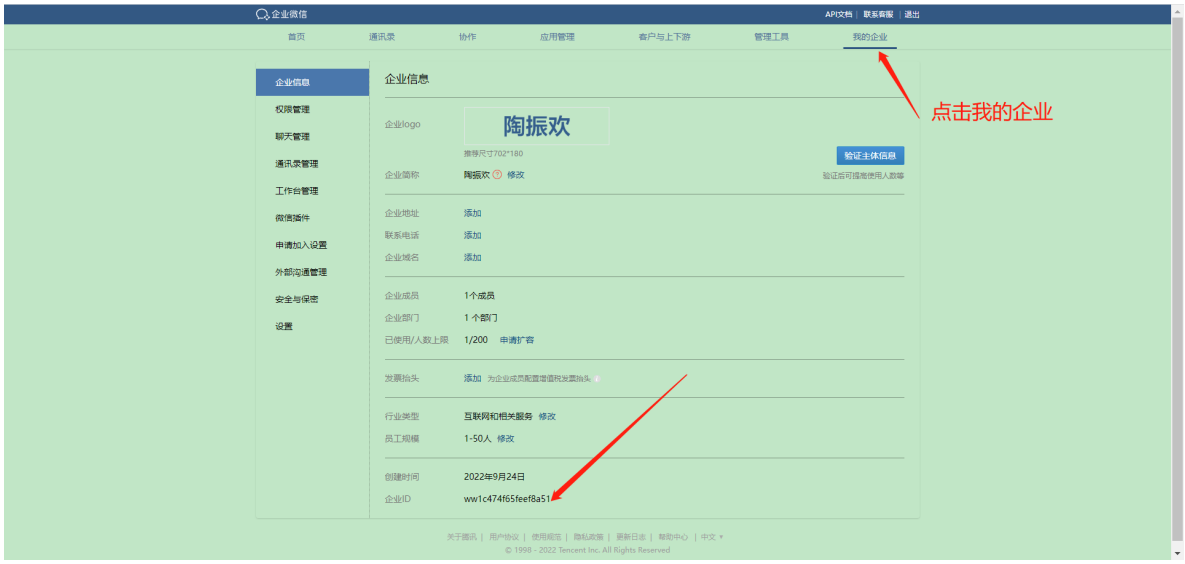
```
[root@k8s-master01 manifests]# kubectl replace -f  alertmanager-secret.yaml
```

3.2、邮件告警通知的基础上新增微信告警通知

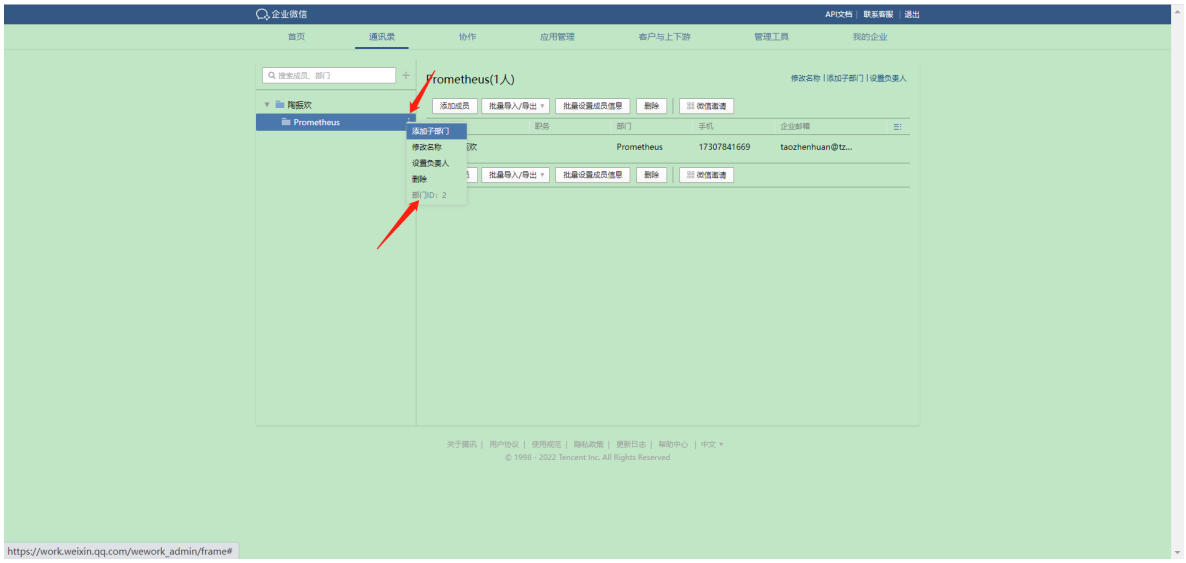
- 登录网页版本企业微信
- 如果公司有账号，用公司的账号即可，但是得有查看以下图片中内容的权限
- 如果没账号可以自己注册一个模拟学习【注册企业账号】

```
https://work.weixin.qq.com/
```

- 需要找到企业ID
    - ww1c474f65feef8a51

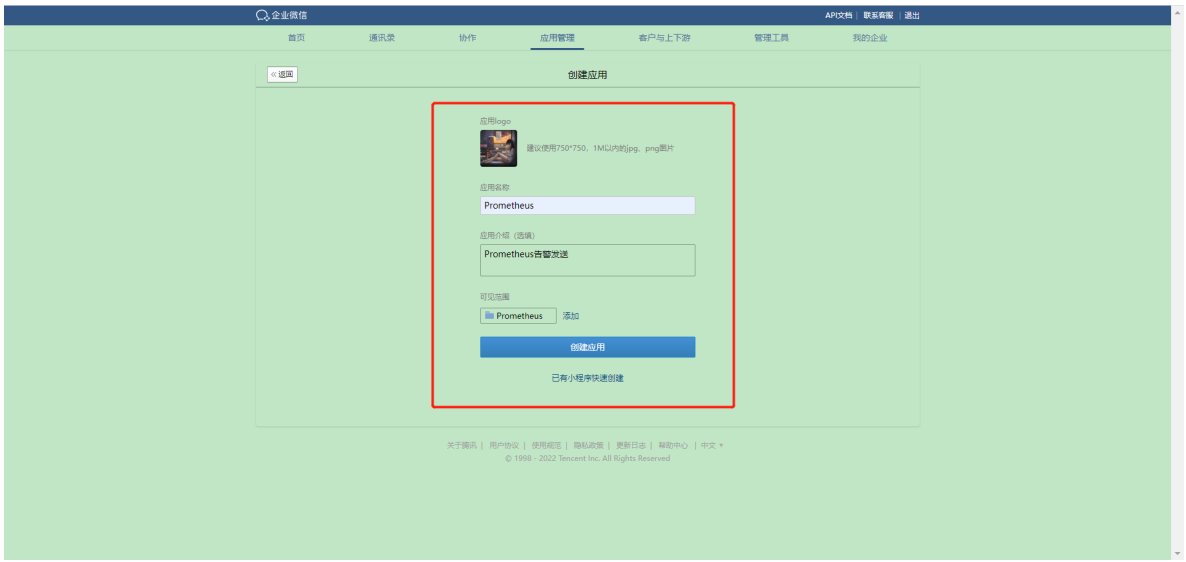


- 创建一个部门用于接收告警信息
- 并且拿到部门ID：2



- 创建应用发布消息

- 输入信息以后点击创建应用即可



- 查看Secret【这需要在Prometheus上配置】
- AgentId：1000002 【需要在用到，在secret上方那个玩意】
- 点击发送后需要在手机上才能看到

- 配置可信 IP（**如果不配置可信 IP 的话，将导致无法发送告警消息**）
  - curl httpbin.org/ip # 获取当前主机的公网 IP

- 编辑配置文件【/root/kube-prometheus/manifests/alertmanager-secret.yaml】，新增企业微信相关信息
- 在邮件信息的下方对齐添加即可

```
# 企业ID：    ww1c474f65feef8a51
# Secret：    _ezOVcpe_dZVyLfy8BWcZbcVX_ex9ADimtnfA7kyHH8
# AgentId：  1000002

  wechat_api_url: 'https://qyapi.weixin.qq.com/cgi-bin/'              #
wechat_api_url是固定的，其他的对应写
  wechat_api_secret: '_ezOVcpe_dZVyLfy8BWcZbcVX_ex9ADimtnfA7kyHH8'  # secret
  wechat_api_corp_id: 'ww1c474f65feef8a51'                           # 企业ID
```

- 配置告警要发生给的部门，部门可以创建多个，按需发送

```
# 收件人配置
"receivers":
    - name: 'wechat'
      wechat_configs:
      - send_resolved: true  # 告警解决后是否发生个告警解决邮件
        to_tag: '2'          # 部门ID
        agent_id: '1000002'  # AgentId
        api_secret: "_ezOVcpe_dZVyLfy8BWcZbcVX_ex9ADimtnfA7kyHH8"   # secret 新版
本加了这参数,老版本估计不需要
```

- 配置route中的告警规则发送人
- Watchdog 更改为wechat，意思是把alertname=Watchdog告警名发给人收件人wechat,不发生给Watchdog,alertname是刚刚配置的receivers

```
"route":
      "group_by":
      - "namespace"
      # 新增2个告警分组规则
      - job
      - alertname
      "group_interval": "5m"
      "group_wait": "30s"
      "receiver": "Default"
      "repeat_interval": "12h"
      "routes":
      - "match":
          "alertname": "Watchdog"
        "receiver": "wechat"  # Watchdog 更改为wechat，意思是把alertname=Watchdog告警名发给人收件人wechat,不发生给Watchdog,alertname是刚刚配置的receivers
```

- 更新配置文件
  - 更新后页面查看是否成功加载配置文件即可 【http://192.168.1.110:30585/#/status】

```
[root@k8s-master01 manifests]# kubectl replace -f  alertmanager-secret.yaml
secret/alertmanager-main replaced
```

四、其他人的

- 【https://blog.csdn.net/weixin_46902396/article/details/125570792】

```
resolve_timeout：在定义的时间内，若没有继续产生告警，才会发送恢复消息；
smtp_require_tls：是否使用 TLS 协议；
group_by：将具有相同属性的告警进行分组聚合；
group_wait：发送告警前的等待时间；
group_interval：发送告警的时间间隔；
repeat_interval：分组内 发送相同告警的时间间隔；
receiver：指定的接收器（和 receivers 配置里的名字匹配）
routes：子路由，通过 匹配告警规则中的标签，来发送到指定的接收人；
send_resolved：是否通知已经解决的告警；

[root@k8s-master01 ~]# cat alertmanager.yml
global:
  resolve_timeout: 3m
  wechat_api_url: "https://qyapi.weixin.qq.com/cgi-bin/"
  wechat_api_secret: "Secret"
  wechat_api_corp_id: "企业微信ID"

templates:
  - '/usr/local/alertmanager/wechat.tmpl'

route:
```

```
  group_by: ['env','instance','type','group','job','alertname']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 1h
  receiver: wechat

receivers:
- name: 'wechat'
  wechat_configs:
  - agent_id: "AgentId"
    to_party: "部门ID"
    message: '{{ template "wechat.default.message" . }}'
    send_resolved: true
```

```
[root@k8s-master01 ~]# cat <<"END" > /usr/local/alertmanager/wechat.tmpl
{{ define "wechat.default.message" }}
{{- if gt (len .Alerts.Firing) 0 -}}
{{- range $index, $alert := .Alerts -}}
{{- if eq $index 0 }}
======== 监控报警 ========
告警状态：{{   .Status }}
告警级别：{{ .Labels.severity }}
告警类型：{{ .Labels.alertname }}
故障主机：{{ .Labels.instance }}
告警主题：{{ .Annotations.summary }}
告警详情：{{ .Annotations.description }};
故障时间：{{ .StartsAt.Format "2006-01-02 15:04:05" }}
======== = END = ========
{{- end }}
{{- end }}
{{- end }}

{{- if gt (len .Alerts.Resolved) 0 -}}
{{- range $index, $alert := .Alerts -}}
{{- if eq $index 0 }}
======== 异常恢复 ========
告警类型：{{ .Labels.alertname }}
告警状态：{{ .Status }}
告警主题：{{ .Annotations.summary }}
告警详情：{{ .Annotations.description }};
故障时间：{{ .StartsAt.Format "2006-01-02 15:04:05" }}
恢复时间：{{ .EndsAt.Format "2006-01-02 15:04:05" }}
{{- if gt (len $alert.Labels.instance) 0 }}
实例信息：{{ $alert.Labels.instance }}
{{- end }}
======== = END = ========
{{- end }}
{{- end }}
{{- end }}
{{- end }}
END
```