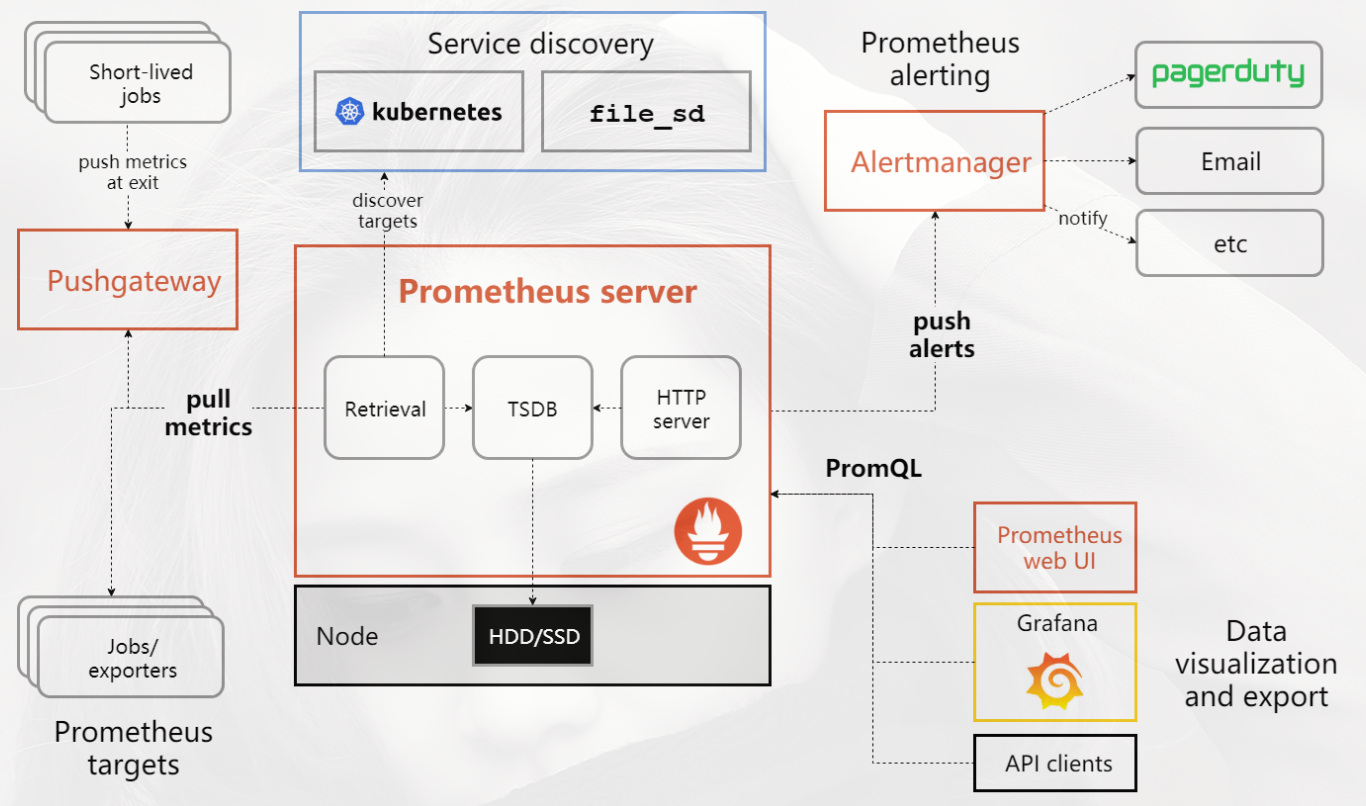
**Prometheus与Grafana入门**

prometheus是由谷歌研发的一款**开源**的监控软件，目前已经被云计算本地基金会托管。其当前相关服务组件如下图所示：



## 普罗米修斯的工作流程

1. **获取数据**：prometheus server根据配置定时去拉取各个节点的数据，默认使用的拉取方式是pull，也可以使用pushgateway提供的push方式获取各个监控节点的数据。
2. **保存数据**： prometheus采集到的监控数据均以metric（指标）形式保存在时序数据库中（TSDB）。对于prometheus获取到的监控数据，可以使用内置的PromQL进行查询，也可以由grafana连接到prometheus进行查询。
3. **报警功能**： prometheus的报警管理功能使用Alertmanager提供。Alertmanager是prometheus的告警管理和发送报警的一个组件，具体告警规则（alert rules）的配置和评估执行由prometheus server完成，prometheus serve将产生的告警事件推送给Alertmanager，再由Alertmanager配置的管理规则来决定如何将通知发送给谁。
4. **展示数据**：prometheus原生的图表功能过于简单，仅用于测试监控配置是否正常还可以，若要用于生产环境大规模指标查询展示，那是远远难以胜任的。一般，可将prometheus数据接入grafana，由grafana进行统一管理。

## Prometheus告警的实现方式

Prometheus的告警能力在其架构中被划分成两个独立的部分。通过在Prometheus中定义AlertRule（告警规则），Prometheus会周期性地对告警规则进行计算，如果满足告警触发条件就会向Alertmanager发送告警信息。告警规则主要由以下几部分组成：告警名称、告警规则、持续时间和标签等。当表达式（PromQL）查询结果持续多长时间后出发告警，可以通过在Prometheus中定义AlertRule（告警规则）来实现。

简单的说，Prometheus将数据采集和告警分成了两个模块。报警规则配置在Prometheus Servers上，然后发送报警信息到AlertManger等告警系统，然后在告警系统管理这些报警信息、聚合报警信息、然后通过email、短信等方式发送消息提。

## Grafana提供的告警功能

Grafana提供了告警配置，且集成了Alertmanager，所以我们可选择使用Grafana来查询监控指标，并使用内部集成的internal Alertmanager(grafana alertmanager)来管理发送告警。Grafana也支持使用prometheus的Alertmanager作为External Alertmanager，

Grafana的Alert页面还可以查看prometheus数据源中配置的告警规则（alert rules），以及将prometheus的Alertmanager作为外部Alertmanager添加进来管理（部分配置项还需在外部Alertmanager直接自行管理，grafana UI中仅可以显示）

## Alertmanager详解

Alertmanager告警通知途径主要使用以下两种：

* 邮件接收器 email\_config
* Webhook接收器 webhook\_config，会用post形式向配置的url地址发送如下格式的参数。

{

"version": "2",

"status": "<resolved|firing>",

"alerts": [{

"labels": < object > ,

"annotations": < object > ,

"startsAt": "<rfc3339>",

"endsAt": "<rfc3339>"

}]

}

Alertmanager除了提供基本的告警通知能力以外，还主要提供了如：分组、抑制以及静默等告警特性：

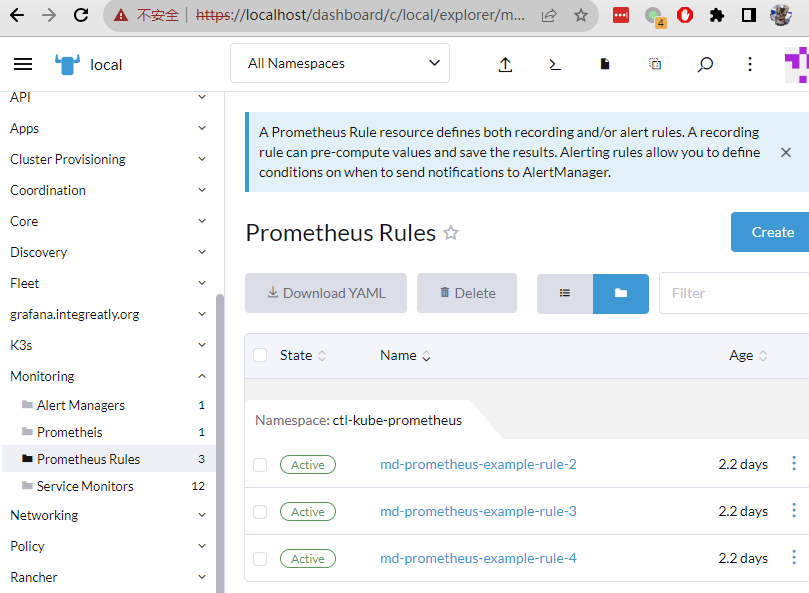


* 分组：分组机制可以将详细的告警信息合并成一个通知。
* 抑制：抑制是指当某一告警发出后，可以停止重复发送由此告警引发的其它告警的机制。
* 静默： 静默提供了一个简单的机制可以快速根据标签对告警进行静默处理。如果接收到的告警符合静默的配置，Alertmanager则不会发送告警通知。

注：grafana internal Alertmanager当前版本暂未支持抑制功能，Prometheus Alertmanager包含上述所有能力。

## 告警方式小结

相比于Grafana的图形化界面，Alertmanager需要依靠配置文件实现，配置比较繁琐。但PrometheusAlertmanager与下面章节介绍的Prometheus Operator结合后，可以使用K8S集群中CRD来配置告警，使用起来也是一个不错的选择，效果图如下所示：



综上所述，可选的实现告警功能主要有以下几种方式：

* 使用prometheus提供的Alertmanager告警组件（功能全面，告警规则配置比较复杂）。
* Grafana 等自带的告警功能（配置简单）
* OneAlert等其他第三方组件（配置简单，可以实现短信、电话、微信等多种告警方式，但是依赖第三方平台，而且是收费的）

## Prometheus Operator

### What is an operator?

Before moving directly to the installation of the Prometheus using the Prometheus Operator, let’s first understand some of the key concepts needed to understand the Prometheus Operator.

Custom Resource Definition (CRD) resource is a way to define your own resource kind like Deployment, StatefulSet etc. CRDs define the structure and validation of the custom kind.

Custom Resource (CR) are the resources that are created by following the structure from a Custom Resource Definition (CRD).

Custom Controller makes sure that our Kubernetes cluster or application always matches its current state with the desired state that we expect it to.

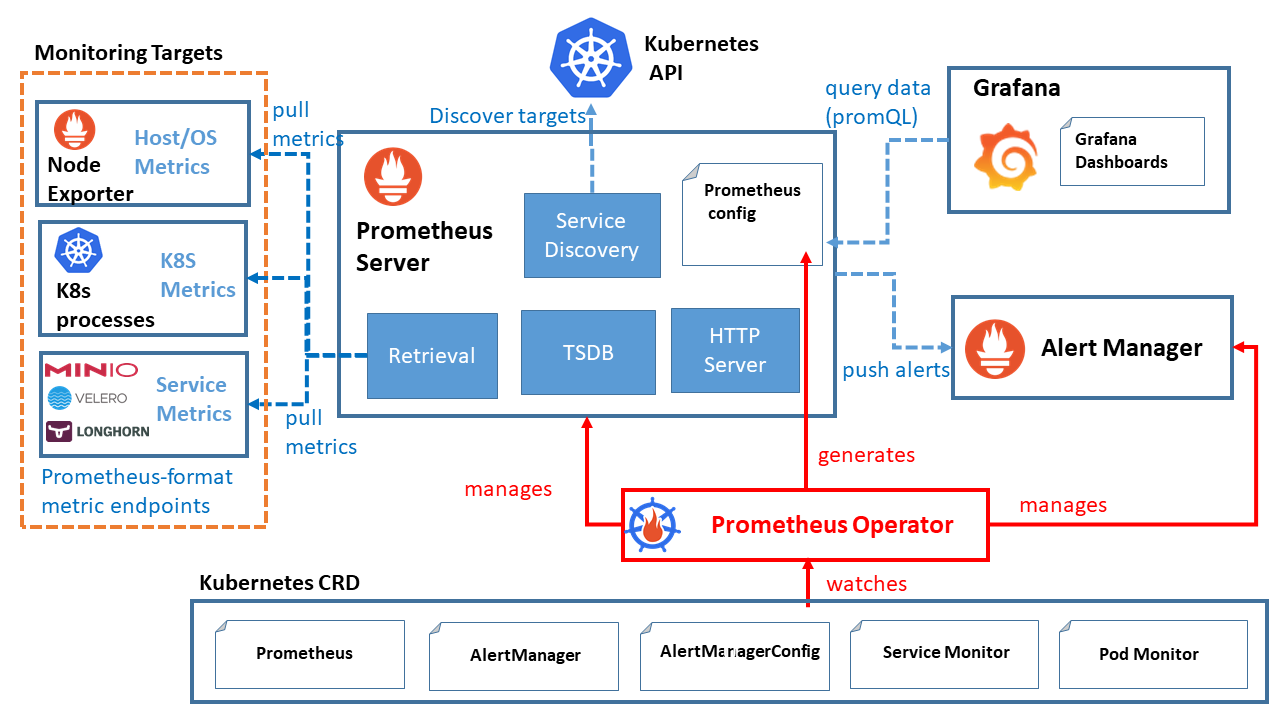
So, Operator is a set of Kubernetes custom controllers that we deploy in the cluster. These listen for changes in the custom resources owned by them (those which we have created using CRDs), and perform certain actions like creating, modifying, deleting Kubernetes resources.

### What is Prometheus operator

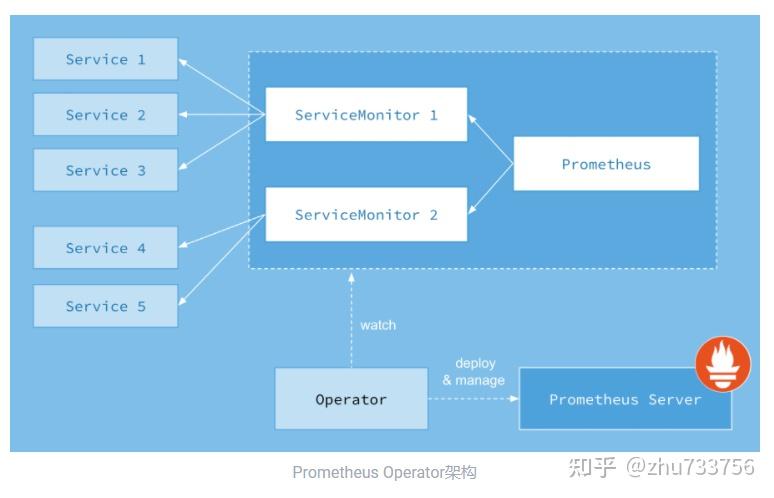
Prometheus has it's own kubernetes operator, which makes it simple to install with helm, and enables users to configure and manage instances of Prometheus using simple declarative configuration that will, in response, create, configure, and manage Prometheus monitoring instances.

Once installed the Prometheus Operator provides the following features:

* Create/Destroy: Easily launch a Prometheus instance for your Kubernetes namespace, a specific application or team easily using the Operator.
* Simple Configuration: Configure the fundamentals of Prometheus like versions, persistence, retention policies, and replicas from a native Kubernetes resource.
* Target Services via Labels: Automatically generate monitoring target configurations based on familiar Kubernetes label queries; no need to learn a Prometheus specific configuration language.



Operator针对管理特定应用程序的，建立在基本的Resource和Controller的概念上，简化图如下：



我们看下这个operator 定义了哪些CRD：

* Prometheus: 定义Prometheus资源实例；
* ServiceMonitor: 定义监控服务；
* PrometheusRule: 告警配置；
* Alertmanager：告警实例；

Reference：

[来不及解释了，快速入门Prometheus（https://zhuanlan.zhihu.com/p/409617025）](https://zhuanlan.zhihu.com/p/409617025)

## Grafana的安装与配置

Grafana安装非常简单，官网下载页面[https://grafana.com/grafana/download](https://grafana.com/grafana/download?pg=graf&plcmt=deploy-box-1)提供多种安装方式的文件及文档。这里我们采用bitnami的[Grafana Operator Chart](https://github.com/bitnami/charts/tree/main/bitnami/grafana-operator)（https://github.com/bitnami/charts/tree/main/bitnami/grafana-operator）来完成grafana在K8S集群中的部署。Grafana Operator的帮助文档请参考https://grafana-operator.github.io/grafana-operator/docs/

### grafana-operator简介

The grafana-operator is a Kubernetes operator built to help you manage your Grafana instances in and outside of Kubernetes.

We make it possible for you to manage and share Grafana dashboards, datasources etc. through code between multiple instances in an easy and scalable way.

[Overview of Grafana Operator](https://github.com/integr8ly/grafana-operator)（<https://github.com/grafana-operator/grafana-operator>）

### grafana-operator安装配置

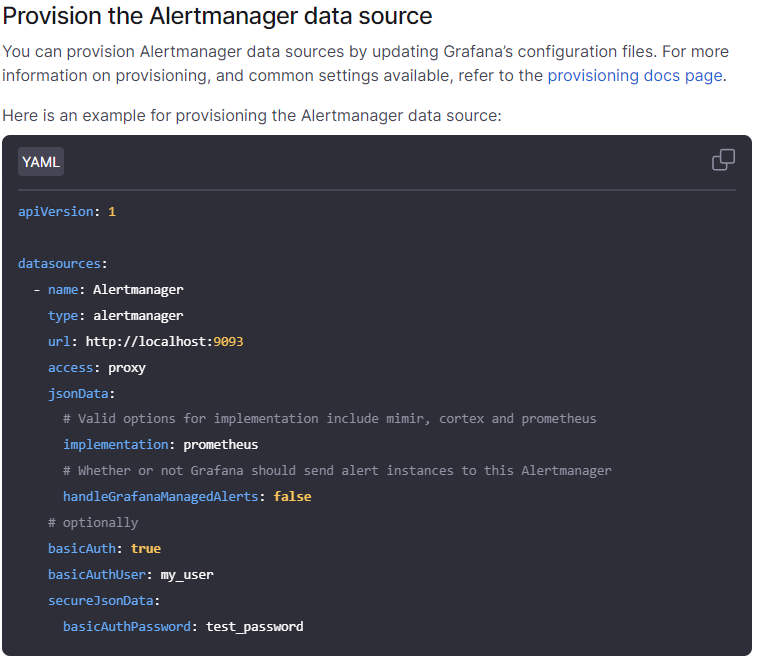
helm chart安装grafana-operator可以选择直接配置grafana许多配置项，例如：管理员密码、邮件通知服务器凭据信息、数据源等信息。

也可以选择使用grafana-operator定义的CRD资源形式在创建grafana后再定义一些配置资源，如数据源信息。此外，数据源信息也可以直接在grafana UI中进行配置和修改。

grafana-operator helm chart安装时添加Prometheus数据源和Prometheus alertmanager数据源信息，样例配置如下：

extraDeploy:  
 - apiVersion: grafana.integreatly.org/v1beta1  
 kind: GrafanaDatasource  
 metadata:  
 name: main-prometheus  
 spec:  
 datasource:  
 access: proxy  
 editable: true  
 isDefault: true  
 jsonData:  
 timeInterval: 30s  
 tlsSkipVerify: true  
 name: prometheus  
 type: prometheus  
 url: >-  
 http://ctl-kube-prometheus-prometheus.ctl-kube-prometheus.svc.cluster.local:9090  
 instanceSelector:  
 matchLabels: {}  
 - apiVersion: grafana.integreatly.org/v1beta1  
 kind: GrafanaDatasource  
 metadata:  
 name: main-prometheus-alertmanager  
 spec:  
 datasource:  
 access: proxy  
 editable: true  
 isDefault: false  
 jsonData:  
 handleGrafanaManagedAlerts: true  
 implementation: prometheus  
 name: prometheus-alertmanager  
 type: alertmanager  
 url: >-  
 http://ctl-kube-prometheus-alertmanager.ctl-kube-prometheus.svc.cluster.local:9093  
 instanceSelector:  
 matchLabels: {}

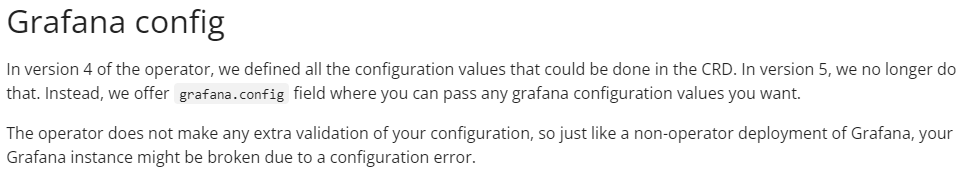
其中，alertmanager的配置项来源于文档<https://grafana.com/docs/grafana/latest/datasources/alertmanager/，文档中yaml>格式配置截图如下：



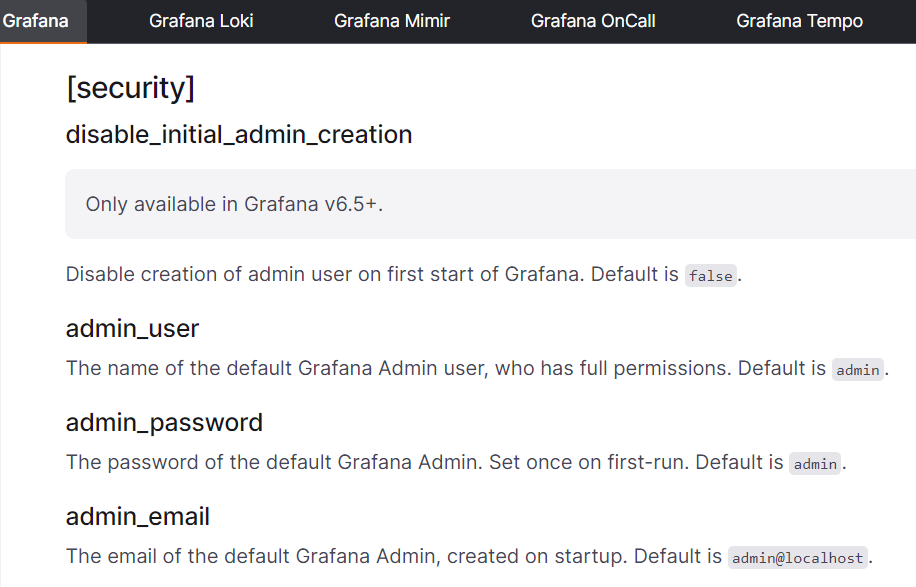
grafana-operator helm chart安装时配置管理员密码配置片和邮件服务器凭据片段如下

grafana:  
 affinity: {}  
 client:  
 timeout: 5  
 config:  
 analytics:  
 check\_for\_updates: 'false'  
 reporting\_enabled: 'false'  
 database:  
 wal: 'true'  
 log:  
 level: warn  
 mode: console  
 security:  
 disable\_gravatar: 'false'  
 admin\_password: adminadminadmin  
 admin\_user: adminadminadmin  
 server:  
 root\_url: >-  
 {{- if .Values.grafana.ingress.enabled }}  
  
 {{ if .Values.grafana.ingress.tls }}https{{ else }}http{{ end }}://{{  
 include "common.tplvalues.render" (dict "value"  
 .Values.grafana.ingress.host "context" $) }}  
  
 {{- else }}  
  
 http://localhost:3000  
  
 {{- end }}  
 auth:  
 disable\_login\_form: 'false'  
 smtp:  
 enabled: 'true'  
 from\_address: zhang.la@asiainfo-sec.com  
 host: smtp.feishu.cn:465  
 password: Yyc\*\*\*\*\*\*awoy  
 user: zhang.la@asiainfo-sec.com

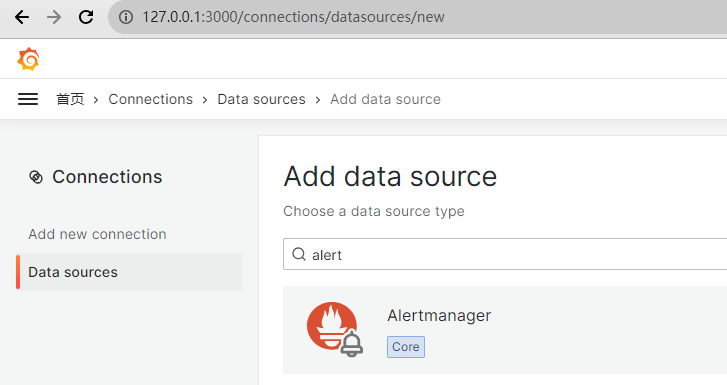
上述配置片段中的grafana.config有许多参数可选，具体感兴趣参数可以直接查看grafana配置文件支持的参数文档<https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/>。下面截图为grafana operator的[Grafana](https://grafana-operator.github.io/grafana-operator/docs/grafana/) config配置选项说明文档截图



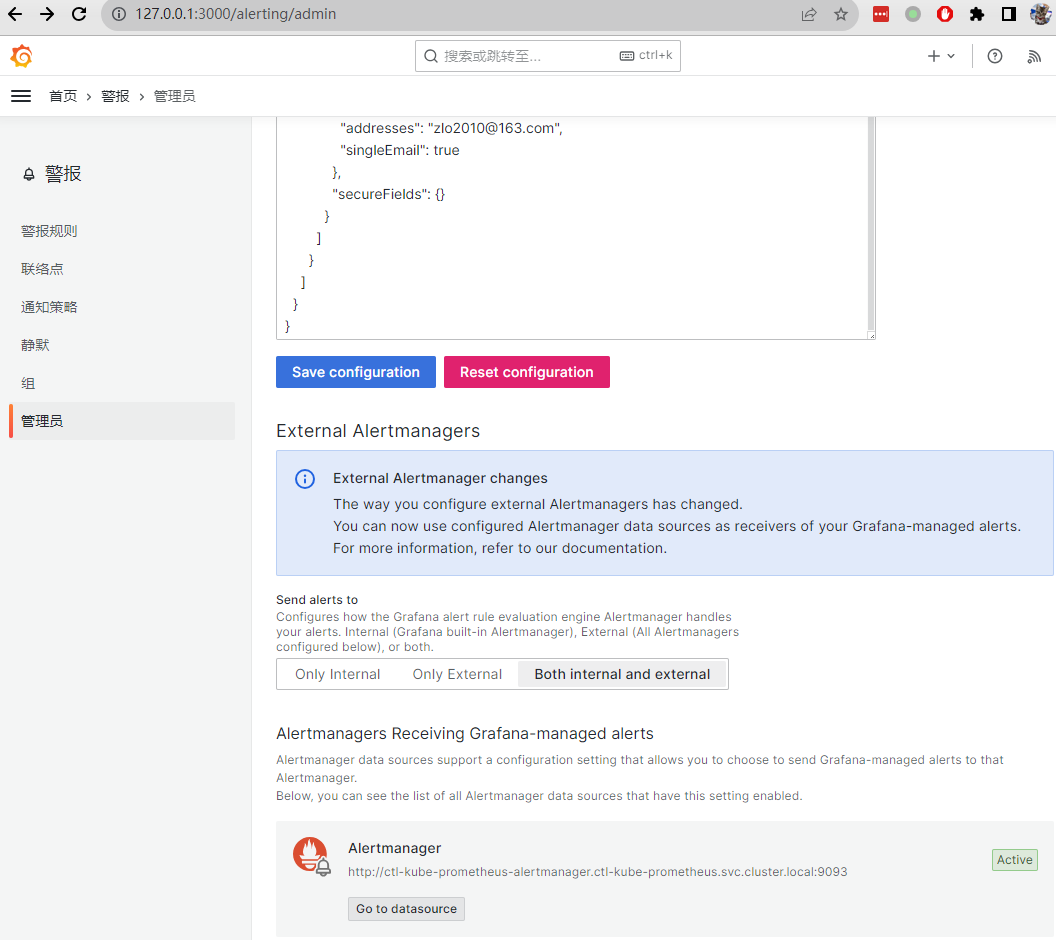
下图为Grafana官网对[Grafana配置文件](https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/" \l "security)支持配置项中管理员登陆配置参数的说明文档（https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/#security）



除在配置文件中直接指定数据源，也可以手动在UI中添加数据源，如下图所示：



添加外部AlertManager后，可以配置Grafana将自生告警规则产生的事件发送至内部Grafana AlertManager还是外部新添加的AlertManager。配置UI截图如下所示：



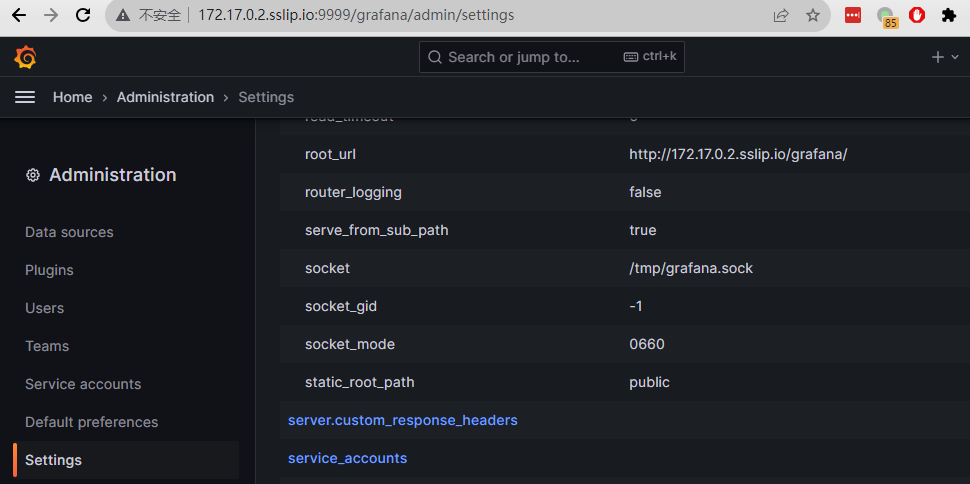
### grafana-operator安装时配置Ingress

grafana-operato安装grafana时，可以指定grafana的ingress。由于Helm Chart可能与我测试的rancher存在一些兼容性问题，以下配置才可以正常安装，否则安装总是失败。

grafana:  
 affinity: {}  
 client:  
 timeout: 5  
 config:  
 server:  
 root\_url: >-  
 {{- if .Values.grafana.ingress.enabled }}  
  
 {{ if .Values.grafana.ingress.tls }}https{{ else }}http{{ end }}://{{  
 .Values.grafana.ingress.host }}{{ .Values.grafana.ingress.path }}  
  
 {{- else }}  
  
 http://localhost:3000  
  
 {{- end }}  
 allow\_embedding: 'true'  
 serve\_from\_sub\_path: 'true'  
 auth:  
 disable\_login\_form: 'false'  
 configMaps: []  
 ingress:  
 annotations: {}  
 enabled: true  
 host: 172.17.0.2.sslip.io  
 ingressClassName: nginx  
 labels: {}  
 path: /grafana/  
 pathType: Prefix

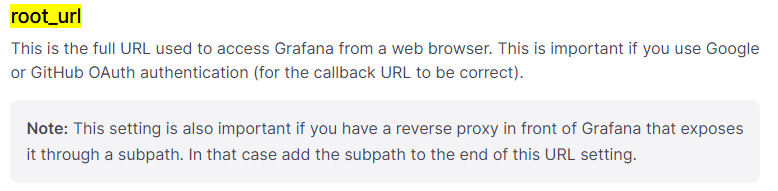
安装完成后，可以通过ingress直接访问。这里本地环境需要先转发一下Ingress Controller服务到本地端口，命令如下，然后再浏览器直接访问grafana。

kubectl --kubeconfig .\local.yaml port-forward --namespace default svc/testingresscontroller-nginx-ingress-controller 9999:80

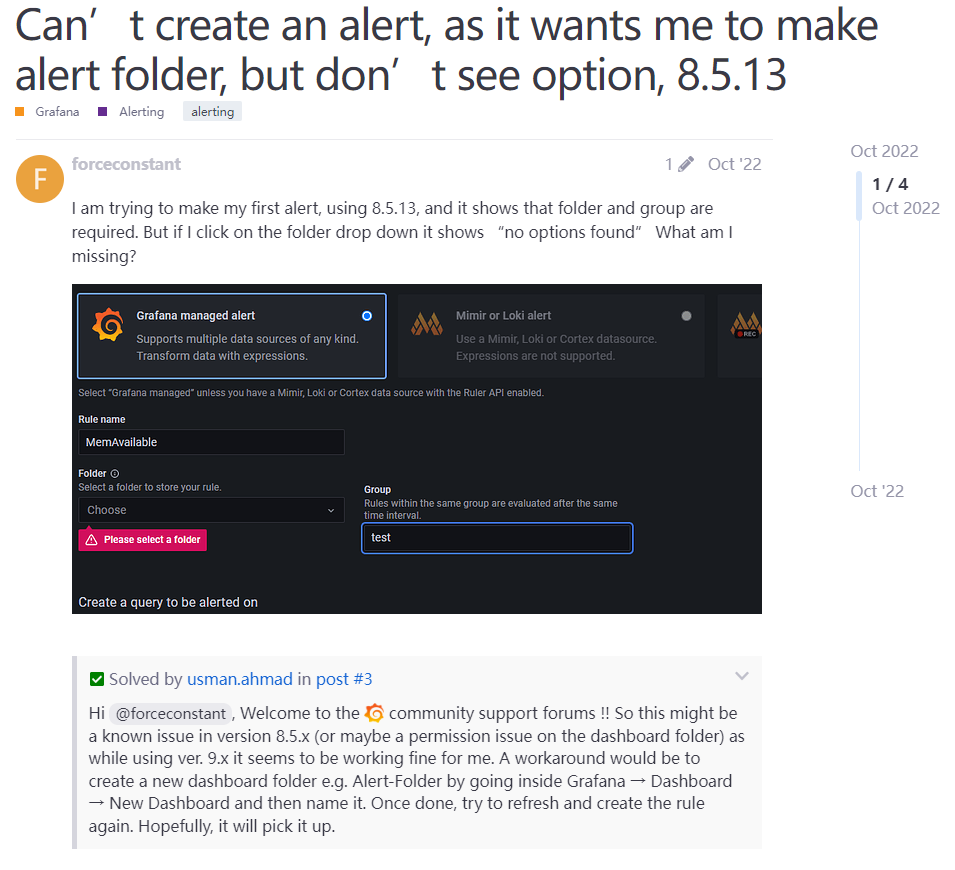


注意：上述配置中root\_url由如下配置项控制，上述配置中片段改为如下配置后

root\_url: >-  
 {{- if .Values.grafana.ingress.enabled }}  
  
 {{ if .Values.grafana.ingress.tls }}https{{ else }}http{{ end }}://{{  
 .Values.grafana.ingress.hostname }}{{ .Values.grafana.ingress.path }}  
  
 {{- else }}  
  
 http://localhost:3000  
  
 {{- end }}



手动在grafana中添加alert rule时，假如出现无法选择Folder的情况，可以选择先去Dashboard中先手动新建目标目录即可，如下截图：



### grafana-operator安装时完整配置

CHART NAME: grafana-operator

CHART VERSION: 3.1.3

APP VERSION: 5.3.0

配置两个datasource、邮件服务器和Ingress有效完整配置文件如下：

clusterDomain: cluster.local  
commonAnnotations: {}  
commonLabels: {}  
extraDeploy:  
 - apiVersion: grafana.integreatly.org/v1beta1  
 kind: GrafanaDatasource  
 metadata:  
 name: main-prometheus  
 spec:  
 datasource:  
 access: proxy  
 editable: true  
 isDefault: true  
 jsonData:  
 timeInterval: 30s  
 tlsSkipVerify: true  
 name: prometheus  
 type: prometheus  
 url: >-  
 http://ctl-kube-prometheus-prometheus.ctl-kube-prometheus.svc.cluster.local:9090  
 instanceSelector:  
 matchLabels: {}  
 - apiVersion: grafana.integreatly.org/v1beta1  
 kind: GrafanaDatasource  
 metadata:  
 name: main-prometheus-alertmanager  
 spec:  
 datasource:  
 access: proxy  
 editable: true  
 isDefault: false  
 jsonData:  
 handleGrafanaManagedAlerts: true  
 implementation: prometheus  
 name: prometheus-alertmanager  
 type: alertmanager  
 url: >-  
 http://ctl-kube-prometheus-alertmanager.ctl-kube-prometheus.svc.cluster.local:9093  
 instanceSelector:  
 matchLabels: {}  
fullnameOverride: ''  
global:  
 imagePullSecrets: []  
 imageRegistry: ''  
grafana:  
 affinity: {}  
 client:  
 timeout: 5  
 config:  
 analytics:  
 check\_for\_updates: 'false'  
 reporting\_enabled: 'false'  
 database:  
 wal: 'true'  
 log:  
 level: warn  
 mode: console  
 security:  
 disable\_gravatar: 'false'  
 admin\_password: zhangla  
 admin\_user: zhangla  
 server:  
 root\_url: >-  
 {{- if .Values.grafana.ingress.enabled }}  
  
 {{ if .Values.grafana.ingress.tls }}https{{ else }}http{{ end }}://{{  
 .Values.grafana.ingress.host }}{{ .Values.grafana.ingress.path }}  
  
 {{- else }}  
  
 http://localhost:3000  
  
 {{- end }}  
 allow\_embedding: 'true'  
 serve\_from\_sub\_path: 'true'  
 auth:  
 disable\_login\_form: 'false'  
 smtp:  
 enabled: 'true'  
 from\_address: zhang.la@asiainfo-sec.com  
 host: smtp.feishu.cn:465  
 password: Yyc\*\*\*\*\*\*\*\*\*\*oy  
 user: zhang.la@asiainfo-sec.com  
 configMaps: []  
 containerSecurityContext:  
 allowPrivilegeEscalation: false  
 enabled: true  
 privileged: false  
 readOnlyRootFilesystem: false  
 runAsGroup: 0  
 runAsNonRoot: true  
 runAsUser: 1001  
 dashboardLabelSelectors:  
 - matchLabels:  
 app.kubernetes.io/instance: '{{ .Release.Name }}'  
 dashboardNamespaceSelector: {}  
 enabled: true  
 envFrom: []  
 extraVolumeMounts: []  
 extraVolumes: []  
 image:  
 digest: ''  
 pullPolicy: IfNotPresent  
 pullSecrets: []  
 registry: docker.io  
 repository: bitnami/grafana  
 tag: 10.0.2-debian-11-r12  
 ingress:  
 annotations: {}  
 enabled: true  
 host: 172.17.0.2.sslip.io  
 ingressClassName: nginx  
 labels: {}  
 path: /grafana/  
 pathType: Prefix  
 jsonnetLibrarySelector:  
 matchLabels:  
 app.kubernetes.io/instance: '{{ .Release.Name }}'  
 labels: {}  
 livenessProbe:  
 enabled: true  
 failureThreshold: 6  
 initialDelaySeconds: 120  
 periodSeconds: 10  
 successThreshold: 1  
 timeoutSeconds: 5  
 nodeAffinityPreset:  
 key: ''  
 type: ''  
 values: []  
 nodeSelector: {}  
 persistence:  
 accessModes:  
 - ReadWriteOnce  
 annotations: {}  
 enabled: false  
 existingVolume: ''  
 size: 10Gi  
 storageClass: ''  
 podAffinityPreset: ''  
 podAntiAffinityPreset: soft  
 podSecurityContext:  
 enabled: true  
 fsGroup: 1001  
 runAsGroup: 0  
 runAsNonRoot: true  
 runAsUser: 1001  
 supplementalGroups: []  
 readinessProbe:  
 enabled: true  
 failureThreshold: 6  
 initialDelaySeconds: 30  
 periodSeconds: 10  
 successThreshold: 1  
 timeoutSeconds: 5  
 replicaCount: 1  
 resources:  
 limits: {}  
 requests: {}  
 secrets: []  
 service:  
 annotations: {}  
 extraPorts: []  
 type: ClusterIP  
 serviceAccount: {}  
 sidecars: []  
 tolerations: []  
 updateStrategy: {}  
kubeVersion: ''  
nameOverride: ''  
namespaceOverride: ''  
operator:  
 affinity: {}  
 args: []  
 command: []  
 containerPorts:  
 metrics: 8080  
 containerSecurityContext:  
 allowPrivilegeEscalation: false  
 enabled: true  
 privileged: false  
 readOnlyRootFilesystem: false  
 runAsGroup: 1001  
 runAsNonRoot: true  
 runAsUser: 1001  
 customLivenessProbe: {}  
 customReadinessProbe: {}  
 customStartupProbe: {}  
 enabled: true  
 extraArgs: []  
 extraEnvVars: []  
 extraEnvVarsCM: ''  
 extraEnvVarsSecret: ''  
 extraVolumeMounts: []  
 extraVolumes: []  
 hostAliases: []  
 image:  
 digest: ''  
 pullPolicy: IfNotPresent  
 pullSecrets: []  
 registry: docker.io  
 repository: bitnami/grafana-operator  
 tag: 5.3.0-debian-11-r0  
 initContainers: []  
 leaderElect: true  
 lifecycleHooks: {}  
 livenessProbe:  
 enabled: true  
 failureThreshold: 3  
 initialDelaySeconds: 10  
 periodSeconds: 10  
 successThreshold: 1  
 timeoutSeconds: 1  
 namespaceScope: false  
 nodeAffinityPreset:  
 key: ''  
 type: ''  
 values: []  
 nodeSelector: {}  
 podAnnotations: {}  
 podAntiAffinityPreset: soft  
 podLabels: {}  
 podSecurityContext:  
 enabled: true  
 fsGroup: 1001  
 runAsGroup: 1001  
 runAsNonRoot: true  
 runAsUser: 1001  
 supplementalGroups: []  
 priorityClassName: ''  
 prometheus:  
 serviceMonitor:  
 enabled: false  
 honorLabels: false  
 interval: ''  
 jobLabel: app.kubernetes.io/name  
 labels: {}  
 metricRelabelings: []  
 namespace: ''  
 relabelings: []  
 scrapeTimeout: ''  
 selector: {}  
 rbac:  
 create: true  
 readinessProbe:  
 enabled: true  
 failureThreshold: 3  
 initialDelaySeconds: 10  
 periodSeconds: 10  
 successThreshold: 1  
 timeoutSeconds: 1  
 replicaCount: 1  
 resources: {}  
 schedulerName: ''  
 serviceAccount:  
 annotations: {}  
 automountServiceAccountToken: true  
 create: true  
 name: ''  
 sidecars: []  
 startupProbe:  
 enabled: true  
 failureThreshold: 3  
 initialDelaySeconds: 10  
 periodSeconds: 10  
 successThreshold: 1  
 timeoutSeconds: 1  
 terminationGracePeriodSeconds: ''  
 tolerations: []  
 topologySpreadConstraints: []  
 updateStrategy:  
 type: Recreate  
 watchNamespace: ''  
 zapDevel: false  
 zapEncoder: ''  
 zapLevel: ''  
 zapSample: ''  
 zapStacktraceLevel: ''  
 zapTimeEncoding: ''

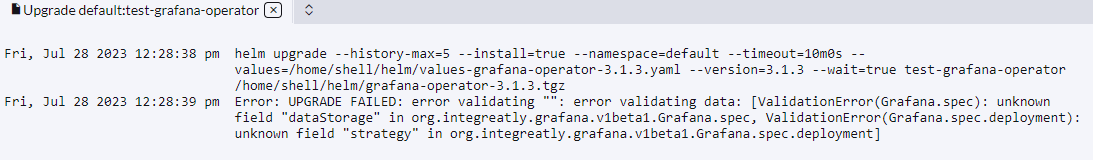
其中，主要改动配置在上述各个小结中均有单独描述，其余均为helm chart默认参数值。

### grafana-operator安装时配置persistence dataStorage

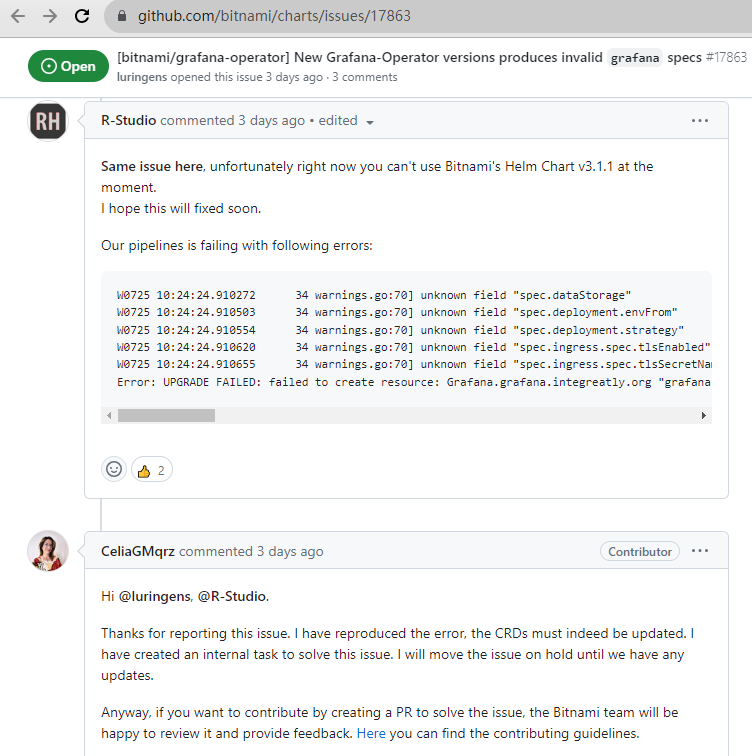
按照Helm Chart如下帮助文档，enable persistence存储时，会出现安装失败的情况



安装失败错误信息如下：



搜索仓库的Issue发现，该问题是Helm Chart新版改动引起的问题，目前还未修复，这里仅测试环境，暂时回退旧版本来使用持久化存储。



### 关联Grafana与Prometheus的方法

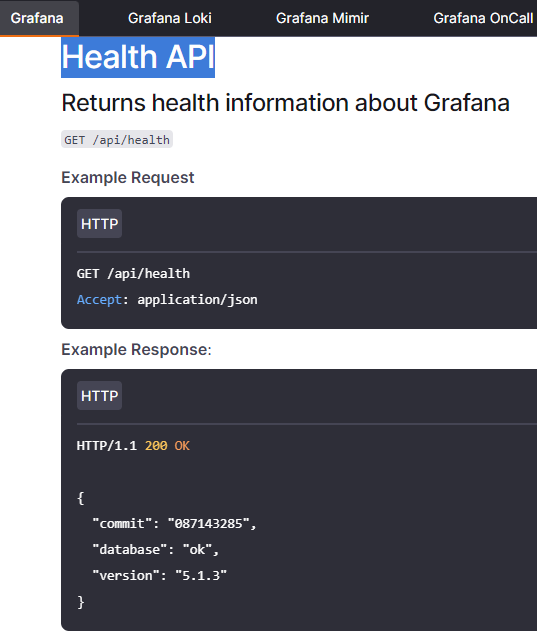
Grafana只是一个优秀的前端展示工具，它自己是不负责数据收集的。所以必须告诉Grafanan如何将Prometheus收集到的数据交给它。方法如下：

*在Grafana的UI界面：*

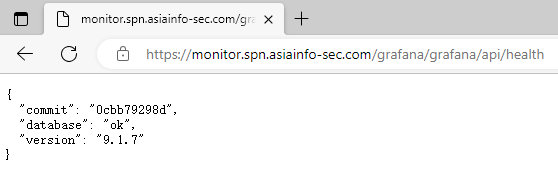
Configuration --> data sources

或者直接在Grafana的配置文件中添加数据源，或者在grafana operator的yaml文件中添加配置片段（上述小结中已经提及相关配置及配置项文档）。

### Grafana service health check API



浏览器查看生产grafana API状态



# Get a Shell to a Running Container

You use [kubectl exec](https://kubernetes.io/docs/tasks/debug/debug-application/get-shell-running-container/) to get a shell to a running container，命令如下：

kubectl --kubeconfig .\local.yaml --namespace default exec --stdin --tty mde-css-a-mde-css-0 -- /bin/sh