



下载APP

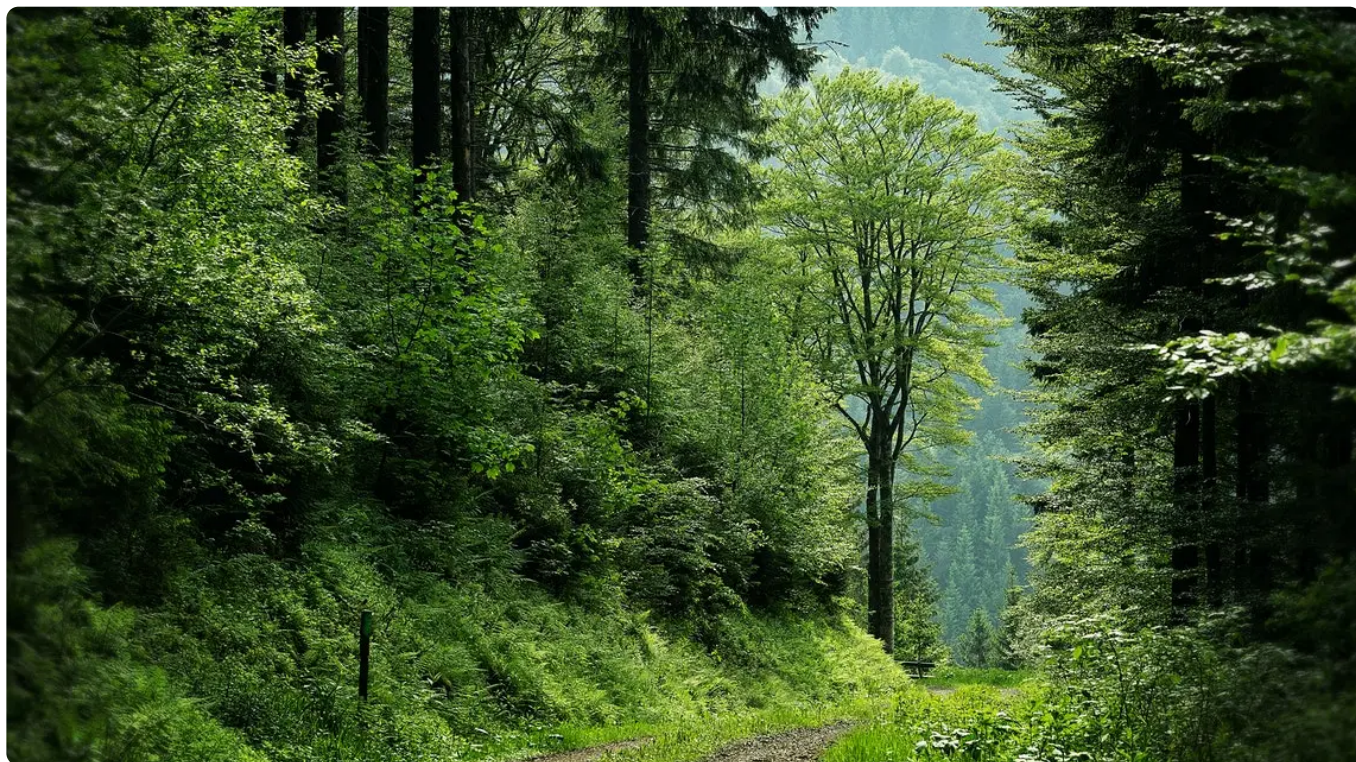


50 | 服务编排（下）：基于Helm的服务编排部署实战

2021-09-25 孔令飞

《Go 语言项目开发实战》

课程介绍 >



讲述：孔令飞

时长 09:00 大小 8.26M



你好，我是孔令飞。

上一讲，我介绍了 Helm 的基础知识，并带着你部署了一个简单的应用。掌握 Helm 的基础知识之后，今天我们就来实战下，一起通过 Helm 部署一个 IAM 应用。

通过 Helm 部署 IAM 应用，首先需要制作 IAM Chart 包，然后通过 Chart 包来一键部署 IAM 应用。在实际开发中，我们需要将应用部署在不同的环境中，所以我也会给你演示下如何在多环境中部署 IAM 应用。



制作 IAM Chart 包

在部署 IAM 应用之前，我们首先需要制作一个 IAM Chart 包。

我们假设 IAM 项目源码根目录为`${IAM_ROOT}`，进入`${IAM_ROOT}/deployments`目录，在该目录下创建 Chart 包。具体创建流程分为四个步骤，下面我来详细介绍下。

第一步，创建一个模板 Chart。

Chart 是一个组织在文件目录中的集合，目录名称就是 Chart 名称（没有版本信息）。你可以看看这个 [🔗 Chart 开发指南](#)，它介绍了如何开发你自己的 Chart。

不过，这里你也可以使用 `helm create` 命令来快速创建一个模板 Chart，并基于该 Chart 进行修改，得到你自己的 Chart。创建命令如下：

```
1 $ helm create iam
```

[📄 复制代码](#)

`helm create iam`会在当前目录下生成一个`iam`目录，里面存放的就是 Chart 文件。Chart 目录结构及文件如下：

```
1 $ tree -FC iam/
2 |— charts/
3 |— Chart.yaml
4 |— templates/
5 |   |— deployment.yaml
6 |   |— _helpers.tpl
7 |   |— hpa.yaml
8 |   |— ingress.yaml
9 |   |— NOTES.txt
10 |   |— serviceaccount.yaml
11 |   |— service.yaml
12 |   |— tests/
13 |       |— test-connection.yaml
14 |— values.yaml
```

[可选]：该目录中放置当前Chart依赖的其他Chart
YAML文件，用于描述Chart的基本信息，包括名称和版本
[可选]：部署文件模版目录，模版使用的值来自values.yaml
Kubernetes Deployment object
用于修改Kubernetes object配置的模板
Kubernetes HPA object
Kubernetes Ingress object
[可选]：放置Chart的使用指南
定义了一些测试资源
Chart的默认配置文件

[📄 复制代码](#)

上面的目录中，有两个比较重要的文件：

Chart.yaml 文件

templates 目录

下面我来详细介绍下这两个文件。**我们先来看 Chart.yaml 文件。**

Chart.yaml 用来描述 Chart 的基本信息，包括名称、版本等，内容如下：

[复制代码](#)

```
1 apiVersion: Chart API 版本（必需）
2 name: Chart名称（必需）
3 version: 语义化版本（必需）
4 kubeVersion: 兼容Kubernetes版本的语义化版本（可选）
5 description: 对这个项目的一句话描述（可选）
6 type: Chart类型（可选）
7 keywords:
8   - 关于项目的一组关键字（可选）
9 home: 项目home页面的URL（可选）
10 sources:
11   - 项目源码的URL列表（可选）
12 dependencies: # chart 必要条件列表（可选）
13   - name: Chart名称 (nginx)
14     version: Chart版本 ("1.2.3")
15     repository: (可选) 仓库URL ("https://example.com/charts") 或别名 ("@repo-name")
16     condition: (可选) 解析为布尔值的YAML路径，用于启用/禁用Chart(e.g. subchart1.enabled)
17     tags: # (可选)
18       - 用于一次启用/禁用 一组Chart的tag
19     import-values: # (可选)
20       - ImportValue 保存源值到导入父键的映射。每项可以是字符串或者一对子/父列表项
21     alias: (可选) Chart中使用的别名。当你要多次添加相同的Chart时会很有用
22 maintainers: # (可选)
23   - name: 维护者名字（每个维护者都需要）
24     email: 维护者邮箱（每个维护者可选）
25     url: 维护者URL（每个维护者可选）
26 icon: 用作icon的SVG或PNG图片URL（可选）
27 appVersion: 包含的应用版本（可选）。不需要是语义化，建议使用引号
28 deprecated: 不被推荐的Chart（可选，布尔值）
29 annotations:
30   example: 按名称输入的批注列表（可选）。
```

我们再来看下templates 目录这个文件。

templates 目录中包含了应用中各个 Kubernetes 资源的 YAML 格式资源定义模板，例如：

[复制代码](#)

```
1 apiVersion: v1
2 kind: Service
```

```

3 metadata:
4   labels:
5     app: {{ .Values.pump.name }}
6     name: {{ .Values.pump.name }}
7 spec:
8   ports:
9     - name: http
10     protocol: TCP
11     {{- toYaml .Values.pump.service.http | nindent 4 }}
12   selector:
13     app: {{ .Values.pump.name }}
14   sessionAffinity: None
15   type: {{ .Values.serviceType }}

```

{{ .Values.pump.name }}会被deployments/iam/values.yaml文件中pump.name的值替换。上面的模版语法扩展了 Go text/template包的语法：

[复制代码](#)

```

1 # 这种方式定义的模版，会去除test模版尾部所有的空行
2 {{- define "test" }}
3 模版内容
4 {{- end }}
5
6 # 去除test模版头部的第一个空行
7 {{- template "test" }}

```

下面是用于 YAML 文件前置空格的语法：

[复制代码](#)

```

1 # 这种方式定义的模版，会去除test模版头部和尾部所有的空行
2 {{- define "test" -}}
3 模版内容
4 {{- end -}}
5
6 # 可以在test模版每一行的头部增加4个空格，用于YAML文件的对齐
7 {{ include "test" | indent 4 }}

```

最后，这里有三点需要你注意：

Chart 名称必须是小写字母和数字，单词之间可以使用横杠-分隔，Chart 名称中不能用大写字母，也不能用下划线，.号也不行。

尽可能使用 [🔗 SemVer 2](#)来表示版本号。

YAML 文件应该按照双空格的形式缩进 (一定不要使用 tab 键)。

第二步，编辑 iam 目录下的 Chart 文件。

我们可以基于helm create生成的模板 Chart 来构建自己的 Chart 包。这里我们添加了创建 iam-apiserver、iam-authz-server、iam-pump、iamctl 服务需要的 YAML 格式的 Kubernetes 资源文件模板：

[📄 复制代码](#)

```
1 $ ls -l iam/templates/*.yaml
2 iam/templates/hpa.yaml                # Kubernetes HPA模板文
3 iam/templates/iam-apiserver-deployment.yaml    # iam-apiserver服务de
4 iam/templates/iam-apiserver-service.yaml       # iam-apiserver服务ser
5 iam/templates/iam-authz-server-deployment.yaml # iam-authz-server服务
6 iam/templates/iam-authz-server-service.yaml    # iam-authz-server服务
7 iam/templates/iamctl-deployment.yaml          # iamctl服务deployment
8 iam/templates/iam-pump-deployment.yaml        # iam-pump服务deployme
9 iam/templates/iam-pump-service.yaml           # iam-pump服务service
```

模板的具体内容，你可以查看 [🔗 deployments/iam/templates/](#)。

在编辑 Chart 时，我们可以通过 helm lint 验证格式是否正确，例如：

[📄 复制代码](#)

```
1 $ helm lint iam
2 ==> Linting iam
3
4 1 chart(s) linted, 0 chart(s) failed
```

0 chart(s) failed 说明当前 iam Chart 包是通过校验的。

第三步，修改 Chart 的配置文件，添加自定义配置。

我们可以编辑deployments/iam/values.yaml文件，定制自己的配置。具体配置你可以参考 [🔗 deployments/iam/values.yaml](#)。

在修改 `values.yaml` 文件时，你可以参考下面这些最佳实践。

变量名称以小写字母开头，单词按驼峰区分，例如 `chickenNoodleSoup`。

给所有字符串类型的值加上引号。

为了避免整数转换问题，将整型存储为字符串更好，并用 `{{ int $value }}` 在模板中将字符串转回整型。

`values.yaml` 中定义的每个属性都应该文档化。文档字符串应该以它要描述的属性开头，并至少给出一句描述。例如：

 复制代码

```
1 # serverHost is the host name for the webserver
2 serverHost: example
3 # serverPort is the HTTP listener port for the webserver
4 serverPort: 9191
```

这里需要注意，所有的 Helm 内置变量都以大写字母开头，以便与用户定义的 `value` 进行区分，例如 `.Release.Name`、`.Capabilities.KubeVersion`。

为了安全，`values.yaml` 中只配置 Kubernetes 资源相关的配置项，例如 Deployment 副本数、Service 端口等。至于 `iam-apiserver`、`iam-authz-server`、`iam-pump`、`iamctl` 组件的配置文件，我们创建单独的 `ConfigMap`，并在 Deployment 中引用。

第四步，打包 Chart，并上传到 Chart 仓库中。

这是一个可选步骤，可以根据你的实际需要来选择。如果想了解具体操作，你可以查看 [🔗 Helm chart 仓库](#) 获取更多信息。

最后，IAM 应用的 Chart 包见 [🔗 deployments/iam](#)。

IAM Chart 部署

上面，我们制作了 IAM 应用的 Chart 包，接下来我们就使用这个 Chart 包来一键创建 IAM 应用。IAM Chart 部署一共分为 10 个步骤，你可以跟着我一步步操作下。

第一步，配置scripts/install/environment.sh。

scripts/install/environment.sh文件中包含了各类自定义配置，你主要配置下面这些跟数据库相关的就可以，其他配置使用默认值。

MariaDB 配置：environment.sh 文件中以MARIADB_开头的变量。

Redis 配置：environment.sh 文件中以REDIS_开头的变量。

MongoDB 配置：environment.sh 文件中以MONGO_开头的变量。

第二步，创建 IAM 应用的配置文件。

[复制代码](#)

```
1 $ cd ${IAM_ROOT}
2 $ make gen.defaultconfigs # 生成iam-apiserver、iam-authz-server、iam-pump、iamct
3 $ make gen.ca # 生成 CA 证书
```

上面的命令会将 IAM 的配置文件存放在目录\${IAM_ROOT}/_output/configs/下。

第三步，创建 iam 命名空间。

我们将 IAM 应用涉及到的各类资源都创建在iam命名空间中。将 IAM 资源创建在独立的命名空间中，不仅方便维护，还可以有效避免影响其他 Kubernetes 资源。

[复制代码](#)

```
1 $ kubectl create namespace iam
```

第四步，将 IAM 各服务的配置文件，以 ConfigMap 资源的形式保存在 Kubernetes 集群中。

[复制代码](#)

```
1 $ kubectl -n iam create configmap iam --from-file=${IAM_ROOT}/_output/configs/
2 $ kubectl -n iam get configmap iam
3 NAME      DATA      AGE
4 iam       4          13s
```

第五步，将 IAM 各服务使用的证书文件，以 ConfigMap 资源的形式保存在 Kubernetes 集群中。

[复制代码](#)

```
1 $ kubectl -n iam create configmap iam-cert --from-file=${IAM_ROOT}/_output/cer
2 $ kubectl -n iam get configmap iam-cert
3 NAME      DATA      AGE
4 iam-cert  14         12s
```

第六步，创建镜像仓库访问密钥。

在准备阶段，我们开通了 [腾讯云镜像仓库服务](#)，并创建了用户10000099``xxxx，密码为iam59!z\$。

接下来，我们就可以创建 docker-registry secret 了。Kubernetes 在下载 Docker 镜像时，需要 docker-registry secret 来进行认证。创建命令如下：

[复制代码](#)

```
1 $ kubectl -n iam create secret docker-registry ccr-registry --docker-server=cc
```

第七步，创建 Docker 镜像，并 Push 到镜像仓库。

[复制代码](#)

```
1 $ make push REGISTRY_PREFIX=ccr.ccs.tencentyun.com/marmotedu VERSION=v1.1.0
```

第八步，安装 IAM Chart 包。

在 [49 讲](#)里，我介绍了 4 种安装 Chart 包的方法。这里，我们通过未打包的 IAM Chart 路径来安装，安装方法如下：

[复制代码](#)

```
1 $ cd ${IAM_ROOT}
2 $ helm -n iam install iam deployments/iam
```



```

3 NAME: iam
4 LAST DEPLOYED: Sat Aug 21 17:46:56 2021
5 NAMESPACE: iam
6 STATUS: deployed
7 REVISION: 1
8 TEST SUITE: None

```

执行 `helm install` 后，Kubernetes 会自动部署应用，等到 IAM 应用的 Pod 都处在 Running 状态时，说明 IAM 应用已经成功安装：

```

1 $ kubectl -n iam get pods | grep iam
2 iam-apiserver-cb4ff955-hs827      1/1      Running    0          66s
3 iam-authz-server-7fccc7db8d-chwnn 1/1      Running    0          66s
4 iam-pump-78b57b4464-rrlbf        1/1      Running    0          66s
5 iamctl-59fdc4995-xrzhn            1/1      Running    0          66s

```

[复制代码](#)

第九步，测试 IAM 应用。

我们通过 `helm install` 在 `iam` 命令空间下创建了一个测试 Deployment `iamctl`。你可以登陆 `iamctl` Deployment 所创建出来的 Pod，执行一些运维操作和冒烟测试。登陆命令如下：

```

1 $ kubectl -n iam exec -it `kubectl -n iam get pods -l app=iamctl | awk '/iamctl

```

[复制代码](#)

登陆到 `iamctl-xxxxxxx-xxxxx` Pod 中后，你就可以执行运维操作和冒烟测试了。

先来看运维操作。 `iamctl` 工具以子命令的方式对外提供功能，你可以使用它提供的各类功能，如下图所示：

```

[root@iamctl-59fdc4995-xrzhn install]# iamctl user list
NAME      NICKNAME  EMAIL          PHONE      CREATED          UPDATED
foo1234   foo1234   aa@qq.com      1812884xxx 2021-06-27 21:18:00 2021-06-27 21:18:01
admin     admin     admin@foxmail.com 1812884xxx 2021-05-27 18:01:40 2021-05-06 05:13:14
[root@iamctl-59fdc4995-xrzhn install]# iamctl secret create foo
secret/foo created
[root@iamctl-59fdc4995-xrzhn install]# iamctl secret list
NAME      SECRETID      SECRETKEY      EXPIRES      CREATED
foo       NQTqB7M0ukhMUMFYoiEdehvd79atnlvUxE3k  IDsj07NX1u09KooH5YJHxdIGyRC8LSBT  2021-09-20 20:31:27 2021-09-14 20:31:27
authzsecret cgdVSe01PFhLTv3TcFEqta9XqBoUdocYUcg0  4j7Bf7zp7YbviSGSZd6bJ5vM7UmoJaTu  1970-01-01 08:00:00 2021-09-14 19:24:42
[root@iamctl-59fdc4995-xrzhn install]#

```

再来看冒烟测试：

[复制代码](#)

```
1 # cd /opt/iam/scripts/install
2 # ./test.sh iam::test::smoke
```

如果./test.sh iam::test::smoke命令打印的输出中，最后一行为congratulations, smoke test passed!字符串，就说明 IAM 应用安装成功。如下图所示：

```
{"metadata":{"id":106,"instanceID":"policy-3lown1","name":"authzpolicy","createdAt":"2021-09-14T20:32:12.237+08:00","updatedAt":"2021-09-14T20:32:12.238+08:00"},"username":"admin","policy":{"id":"","description":"One policy to rule them all.","subjects":["users:\u003cpeter\u003e","users:maria","groups:admins"],"effect":"allow","resources":["resources:articles:\u003c*\u003e","resources:printer"],"actions":["delete","\u003ccreate|update\u003e"],"conditions":{"remoteIPAddress":{"type":"CIDRCondition","options":{"cidr":"192.168.0.1/16"}}},"meta":null}}
null
congratulations, /v1/authz test passed!
congratulations, iam-authz-server test passed!
congratulations, iam-pump test passed!
congratulations, iamctl test passed!
congratulations, smoke test passed!
```

第十步，销毁 EKS 集群的资源。

[复制代码](#)

```
1 $ kubectl delete namespace iam
```

你可以根据需要选择是否删除 EKS 集群，如果不需要了就可以选择删除。

IAM 应用多环境部署

在实际的项目开发中，我们需要将 IAM 应用部署到不同的环境中，不同环境的配置文件是不同的，那么 IAM 项目是如何进行多环境部署的呢？

IAM 项目在 [configs](#) 目录下创建了多个 Helm values 文件（格式为values-{envName}-env.yaml）：

values-test-env.yaml，测试环境 Helm values 文件。

values-pre-env.yaml，预发环境 Helm values 文件。

values-prod-env.yaml，生产环境 Helm values 文件。

在部署 IAM 应用时，我们在命令行指定-f参数，例如：

[复制代码](#)

```
1 $ helm -n iam install -f configs/values-test-env.yaml iam deployments/iam # 安
```

总结

这一讲，我们通过 `helm create iam` 创建了一个模板 Chart，并基于这个模板 Chart 包进行了二次开发，最终创建了 IAM 应用的 Helm Chart 包：[deployments/iam](#)。

有了 Helm Chart 包，我们就可以通过 `helm -n iam install iam deployments/iam` 命令来一键部署好整个 IAM 应用。当 IAM 应用中的所有 Pod 都处在 Running 状态后，说明 IAM 应用被成功部署。

最后，我们可以登录 `iamctl` 容器，执行 `test.sh iam::test::smoke` 命令，来对 IAM 应用进行冒烟测试。

课后练习


1. 试着在 Helm Chart 中加入 MariaDB、MongoDB、Redis 模板，通过 Helm 一键部署好整个 IAM 应用。
2. 试着通过 `helm` 命令升级、回滚和删除 IAM 应用。

欢迎你在留言区与我交流讨论，我们下一讲见。

分享给需要的人，Ta订阅后你可得 **24** 元现金奖励

 生成海报并分享

 赞 0

 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

更多学习推荐

175 道 Go 工程师 大厂常考面试题

限量免费领取 

精选留言 (1)

[写留言](#)**我来也**

2021-09-25

“给所有字符串类型的值加上引号。”

深有体会，很多开源的chart也可能存在这种问题。比如pvc的名称，没有用quote加引号，用户如果非得来一个全数字的pvc就悲剧了。helm会默认转换为数字类型。

我一般调试时使用helm upgrade --install --debug --dry-run。如果可以看到渲染后的ya...
展开

作者回复：好办法！

