K8s管理系统项目实战【前端开发】

讲师：杜Sir

阿良教育：www.aliangedu.cn

一、项目概述

本节课程是k8s管理系统项目实战的前端开发部分，在完成API接口的整体开发后，我们开始着手于前端部分，构建一个个功能页面，将管理系统平台化。

前端部分使用vue3框架以及element-plus组件完成，开发过程中，我们会使用到以下依赖：

（1）xterm 命令行终端模拟器

（2）nprogress 浏览器顶部的进度条

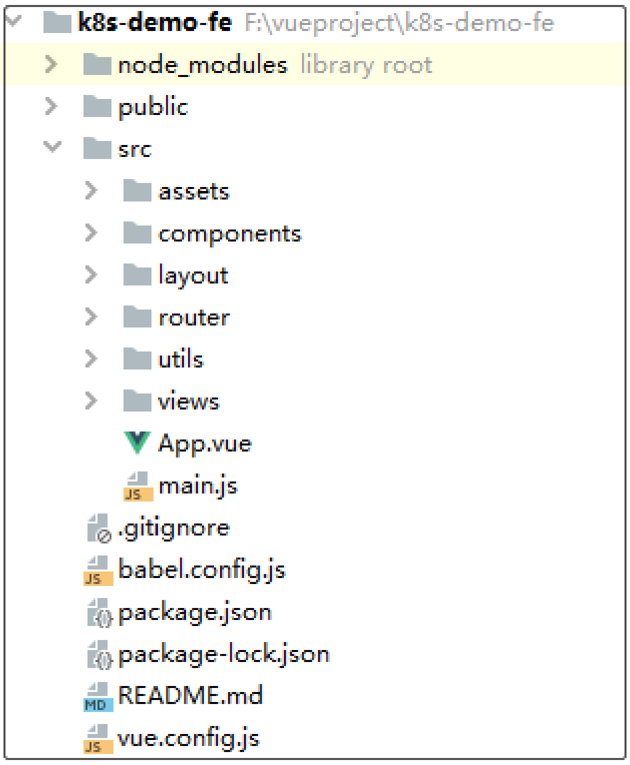
（3）jsonwebtoken jwt token的生成与校验组件

（4）json-editor-vue3/codemirror-editor-vue3 代码编辑器，用于编辑k8s资源YAML

（5）echarts 画图组件，如柱状图、饼图等

二、Vue目录结构及启动

1、目录结构



```
k8s-demo-fe F:\vueproject\k8s-demo-fe
  node_modules library root
  public
  src
    assets
    components
    layout
    router
    utils
    views
    App.vue
    main.js
  .gitignore
  babel.config.js
  package.json
  package-lock.json
  README.md
  vue.config.js
```

node_modules：存放npm下载的依赖包

public：站点图标和主页

package.json/package-lock.json：存放依赖版本及项目描述信息

babel.config.js: babel的配置文件，babel是js的编译器

vue.config.js：vue的配置文件

src/下：

views/common/Config.js: 存放后端接口路径、编辑器配置等公共属性

assets：存放图片等静态资源

components：存放自定义的公共组件

layout：存放布局视图文件

router：定义路由配置及规则

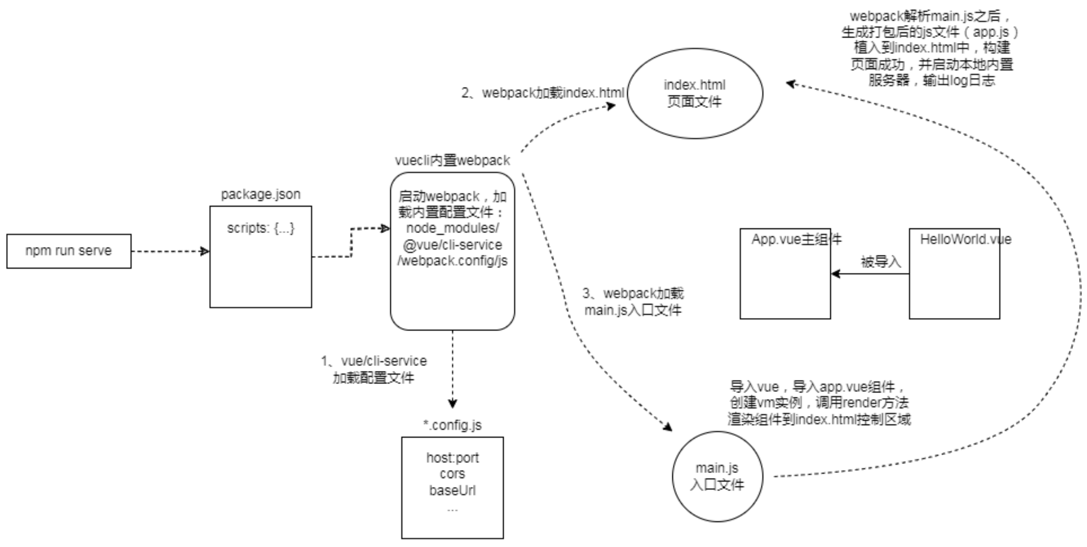
utils：工具类，用于常用方法的封装

views：存放各个页面的视图文件

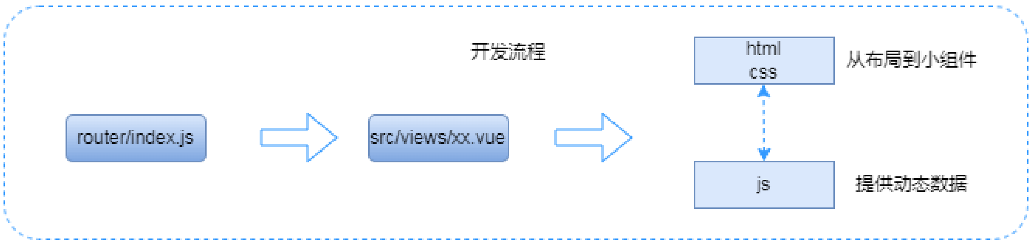
App.vue：主组件，所有页面都是在App.vue下进行切换，可以理解为所有的路由都是App.vue的子组件

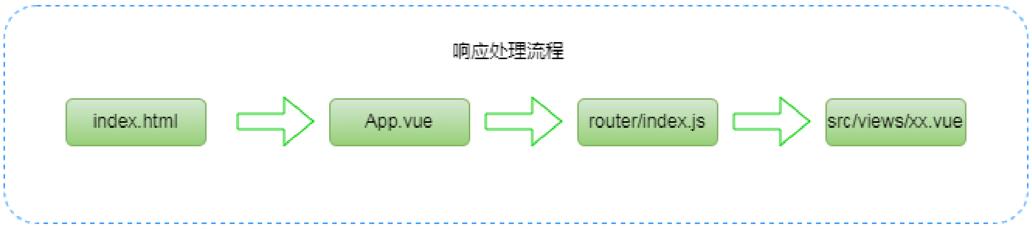
main.js：入口文件，主要作用是初始化vue实例，并引入所需插件

2、启动过程


webpack解析main.js之后，生成打包后的js文件（app.js）植入到index.html中，构建页面成功，并启动本地内置服务器，输出log日志
2、webpack加载index.html
index.html 页面文件
vuecli内置webpack
启动webpack，加载内置配置文件：node_modules/@vue/cli-service/webpack.config/js
package.json
scripts: {...}
npm run serve
App.vue主组件
被导入
HelloWorld.vue
3、webpack加载main.js入口文件
1、vue/cli-service加载配置文件
*.config.js
host:port
cors
baseUrl
...
导入vue，导入app.vue组件，创建vm实例，调用render方法渲染组件到index.html控制区域
main.js 入口文件

三、开发&响应流程





四、框架搭建

1、初始化Vue项目

（1）创建vue3项目

```
vue create k8s-platform-fe
```

（2）关闭语法检查配置文件，关闭语法检测，设置端口号

vue.config.js

```
const { defineConfig } = require('@vue/cli-service')
module.exports = defineConfig({
  devServer:{
    host: '0.0.0.0',//监听地址
    port: 7070, // 启动端口号
    open: true  // 启动后是否自动打开网页
  },
  transpileDependencies: true,
  //关闭语法检测
  lintOnSave: false
})
```

（3）初始化main.js以及安装插件

main.js

```
import { createApp } from 'vue'
//引入element plus
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'
//引入图标视图
```

```
import * as ELIcons from '@element-plus/icons-vue'
//引入App.vue主组件
import App from './App.vue'
//引入路由配置及规则
import router from './router'

//创建vue实例
const app = createApp(App)
//将图标注册为全局组件
for (let iconName in ELIcons) {
    app.component(iconName, ELIcons[iconName])
}
//引入element plus
app.use(ElementPlus)
//引入路由
app.use(router)
//挂载
app.mount('#app')
```

（4）初始化App.vue

```
<template>
    <span>我是App.vue</span>
    <!-- 路由占位符，会导入匹配到的$route.path的视图组件 -->
    <router-view></router-view>
</template>

<style>
    /*设置html和body*/
    html, body {
        width: 100%;
        height: 100%;
        padding: 0;
        margin: 0;
    }
    #nprogress .bar {
        /*自定义进度条颜色*/
        background: #2186c0 !important;
    }
</style>
```

2、封装路由

src/views/home/Home.vue

```
<template>
    <div class="home">
        我是Home.vue
    </div>
</template>
```

router/index.js

```
//导入router的路由模式
import {createRouter, createWebHistory} from 'vue-router'
```

```
//路由规则
const routes = [
    {
        path: '/home',
        name: '概要',
        icon: 'odometer',
        meta: {titale: "概要", requireAuth: true},
        component: () => import('@/views/home/Home.vue')
    },
]

//创建路由实例
const router = createRouter({
    //hash模式：createWebHashHistory
    //history模式：createWebHistory
    history: createWebHistory(),
    routes
})
//抛出路由实例，在main.js中引用
export default router
```

3、添加进度条

router/index.js

```
//导入进度条组件
import NProgress from 'nprogress'
import 'nprogress/nprogress.css'

//递增进度条，这将获取当前状态值并添加0.2直到状态为0.994
NProgress.inc(100)
//easing 动画字符串
//speed 动画速度
//showSpinner 进度环显示隐藏
NProgress.configure({ easing: 'ease', speed: 600, showSpinner: false })

//router.beforeEach（）一般用来做一些进入页面的限制。比如没有登录，就不能进入某些
//页面，只有登录了之后才有权限查看某些页面。。。说白了就是路由拦截。
//to 要去到某个页面的属性
//from 从哪个页面来的属性
//next 处理路由跳转及放行
router.beforeEach((to, from, next) => {
    // 启动进度条
    NProgress.start()

    // 设置头部
    if (to.meta.title) {
        document.title = to.meta.title
    } else {
        document.title = "Kubernetes"
    }
    //放行
    next()
})

router.afterEach(() => {
    // 关闭进度条
    NProgress.done()
})
```

4、启动/测试

```
npm run serve
```

5、封装axios

封装axios请求，添加自定义配置，如超时、重试、header等等

utils/request.js

```
import axios from 'axios';

//新建个axios对象
const httpClient = axios.create({
    validateStatus(status) {
        return status >= 200 && status <= 504 // 设置默认的合法的状态,若状态码不合法，则不会接收response
    },
    timeout: 10000   //超时时间10秒
});


httpClient.defaults.retry = 3 // 请求重试次数
httpClient.defaults.retryDelay = 1000 // 请求重试时间间隔
httpClient.defaults.shouldRetry = true // 是否重试

//添加请求拦截器
httpClient.interceptors.request.use(
    config => {
        //添加header
        config.headers['Content-Type'] = 'application/json'
        config.headers['Accept-Language'] = 'zh-CN'
        config.headers['Authorization'] = localStorage.getItem('token') // 可以全局设置接口请求header中带token

        if (config.method === 'post') {
            if (!config.data) { // 没有参数时，config.data为null，需要转下类型
                config.data = {}
            }
        }
        return config
    },
    err => {
        //Promise.reject()方法返回一个带有拒绝原因的Promise对象，在F12的console中显示报错
        Promise.reject(err)
    }
);

//添加响应拦截器
httpClient.interceptors.response.use(
    response => {
        if (response.status !== 200) {
            return Promise.reject(response.data)
        } else {
            return response.data
        }
    },
    err => {
        return Promise.reject(err)
    }
);

export default httpClient;
```

6、处理404页面

（1）404页面

common/404.vue

```
<template>
    <div class="main-body-div">
        <el-row>
            <!-- 图片 -->
            <el-col :span="24">
                <div>
                    <img class="main-body-img" src="../../assets/img/404.png" />
                </div>
                </el-col>
            <!-- 描述 -->
            <el-col :span="24">
                <div>
                    <p class="status-code">404</p>
                    <p class="status-describe">你所访问的页面不存在······</p>
                </div>
            </el-col>
        </el-row>
    </div>
</template>

<script>
export default {
}
</script>


<style scoped>
  /* 图片属性 */
  .main-body-img {
    margin-top: 150px
  }
  /* 整体位置 */
  .main-body-div {
    text-align: center;
    height: 100vh;
    width: 100vw;
  }
  /* 状态码 */
  .status-code {
    margin-top: 20px;
    margin-bottom: 10px;
    font-size: 95px;
    font-weight: bold;
    color: rgb(54, 95, 230);
  }
  /* 描述 */
  .status-describe {
    color: rgb(145, 143, 143);
  }
</style>
```

（2）403页面

common/403.vue

```
<template>
    <div class="main-body-div">
        <el-row>
            <!-- 图片 -->
            <el-col :span="24">
                <div>
                    <img class="main-body-img" src="../../assets/img/403.png" />
                </div>
                </el-col>
            <el-col :span="24">
                <!-- 描述 -->
                <div>
                    <p class="status-code">403</p>
                    <p class="status-describe">你暂时无权限访问该页面······</p>
                </div>
            </el-col>
        </el-row>
    </div>
</template>

<script>
export default {
}
</script>


<style scoped>
  /* 图片属性 */
  .main-body-img {
    margin-top: 15%
  }
  /* 整体位置 */
  .main-body-div {
    text-align: center;
    height: 100vh;
    width: 100vw;
  }
  /* 状态码 */
  .status-code {
    margin: 20px 0 20px 0;
    font-size: 95px;
    font-weight: bold;
    color: rgb(54, 95, 230);
  }
  /* 描述 */
  .status-describe {
    color: rgb(145, 143, 143);
  }
</style>
```

（3）404路由规则

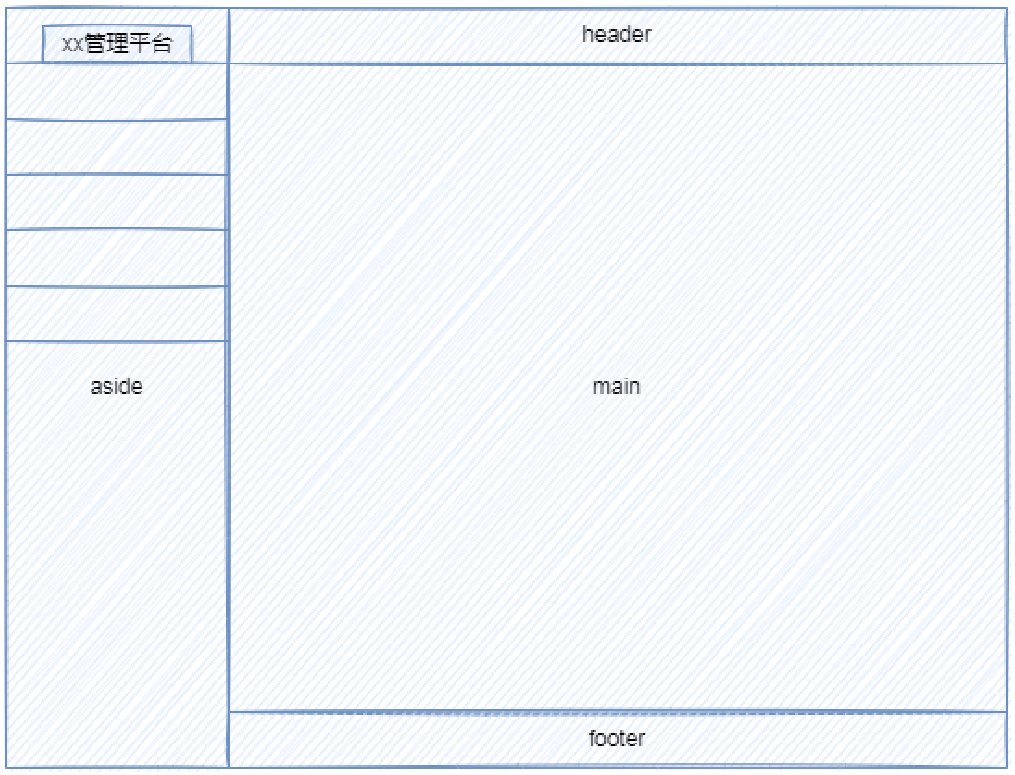
router/index.js的routes变量中增加

```
    {
        path: '/404',
        component: () => import('@/views/common/404.vue'),
        meta: {
            title: '404'
        }
    },
    {
        path: '/403',
        component: () => import('@/views/common/403.vue'),
        meta: {
            title: '403'
        }
    },
    //其他路径跳转至404页面
    {
        path: '/:pathMatch(.*)',
        redirect: '/404'
    },
```

五、前端开发

> 阿良教育： www.aliangedu.cn

1、整体布局



(1）Container布局框架

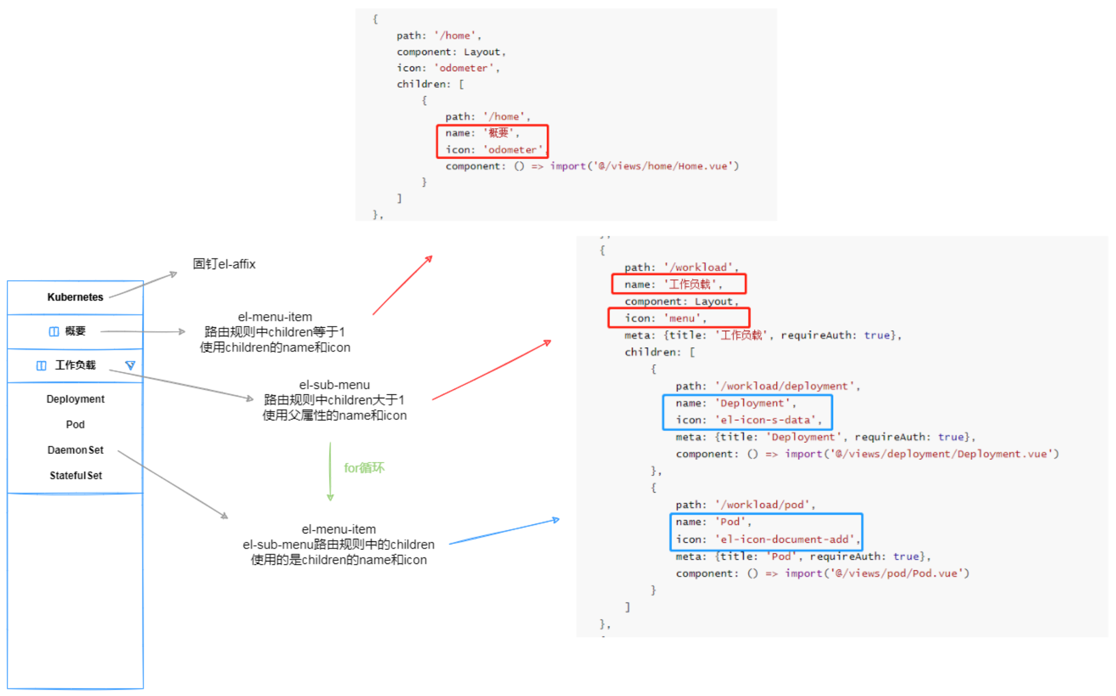
```
<template>
    <div class="common-layout">
        <el-container>
            <el-aside width="200">Aside</el-aside>
            <el-container>
                <el-header>Header</el-header>
                <el-main>Main</el-main>
                <el-footer>Footer</el-footer>
            </el-container>
        </el-container>
    </div>
</template>
```

(2）添加路由规则

```
const routes = [
    {
        path: '/home',
        component: Layout,
        icon: 'odometer',
        children: [
            {
                path: '/home',
                name: '概要',
                icon: 'odometer',
                component: () => import('@/views/home/Home.vue')
            }
        ]
    },
    {
        path: '/workload',
        name: '工作负载',
        component: Layout,
        icon: 'menu',
        meta: {title: '工作负载', requireAuth: true},
        children: [
            {
                path: '/workload/deployment',
                name: 'Deployment',
                icon: 'el-icon-s-data',
                meta: {title: 'Deployment', requireAuth: true},
                component: () => import('@/views/deployment/Deployment.vue')
            },
            {
                path: '/workload/pod',
                name: 'Pod',
                icon: 'el-icon-document-add',
                meta: {title: 'Pod', requireAuth: true},
                component: () => import('@/views/pod/Pod.vue')
            }
        ]
    },
    {
        path: '/404',
        component: () => import('@/views/common/404.vue'),
        meta: {titale: "404", requireAuth: true},
    },
    //其他路径跳转至404页面
    {
        path: '/:pathMatch(.*)',
        redirect: '/404'
    }
]
```

（3）菜单导航栏

功能：固钉、vue-router模式的menu、折叠



```
{
    path: '/home',
    component: Layout,
    icon: 'odometer',
    children: [
        {
            path: '/home',
            name: '概要',
            icon: 'odometer',
            component: () => import('@/views/home/Home.vue')
        }
    ]
},
```

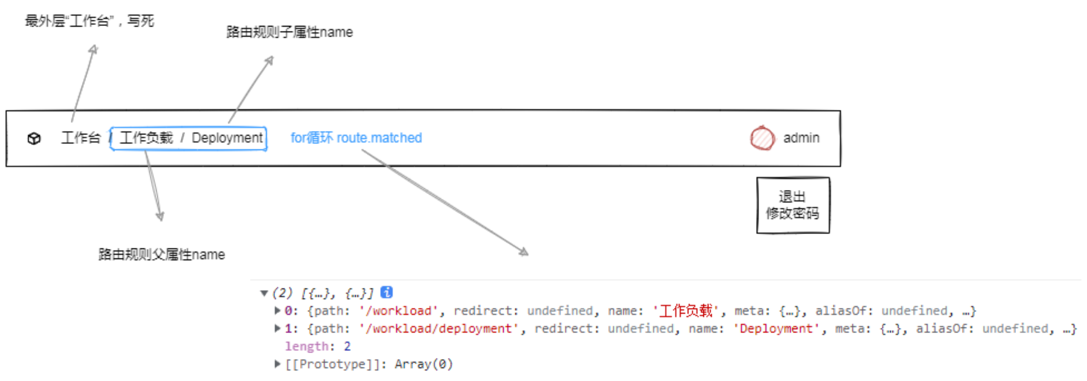
```
{
    path: '/workload',
    name: '工作负载',
    component: Layout,
    icon: 'menu',
    meta: {title: '工作负载', requireAuth: true},
    children: [
        {
            path: '/workload/deployment',
            name: 'Deployment',
            icon: 'el-icon-s-data',
            meta: {title: 'Deployment', requireAuth: true},
            component: () => import('@/views/deployment/Deployment.vue')
        },
        {
            path: '/workload/pod',
            name: 'Pod',
            icon: 'el-icon-document-add',
            meta: {title: 'Pod', requireAuth: true},
            component: () => import('@/views/pod/Pod.vue')
        }
    ]
},
```

```
<template>
    <div class="common-layout">
        <!-- container整体布局 -->
        <el-container style="height: 100vh;">
            <!-- 侧边栏，定义默认宽度 -->
            <el-aside class="aside" :width="asideWidth">
                <!-- 固钉，将平台logo和名字固钉在侧边栏最上方 -->
                <!-- z-index是显示优先级 -->
                <el-affix class="aside-affix" :z-index="1200">
                    <div class="aside-logo">
                        <!-- logo图片 -->
                        <el-image class="logo-image" :src="logo" />
                        <!-- 平台名，折叠后不显示 -->
                        <span :class="[isCollapse ? 'is-collapse' : '']">
                            <span class="logo-name">Kubernetes</span>
                        </span>
                    </div>
                </el-affix>
                <!-- 菜单导航栏 -->
                <!-- router 使用 vue-router 的模式，启用该模式会在激活导航时以 index 作为
path 进行路由跳转 -->
                <!-- default-active 当前激活菜单的index,将菜单栏与路径做了对应关系 -->
                <!-- collapse 是否折叠 -->
                <el-menu class="aside-menu"
                    router
                    :default-active="$route.path"
                    :collapse="isCollapse"
                    background-color="#131b27"
                    text-color="#bfcbd9"
                    active-text-color="#20a0ff">
                    <!-- for循环路由规则 -->
                    <div v-for="menu in routers" :key="menu">
                        <!-- 处理子路由只有1个的情况，如概要、工作流 -->
                        <el-menu-item class="aside-menu-item" v-if="menu.children &&
menu.children.length == 1" :index="menu.children[0].path">
```

```
                            <!-- 引入图标的方式 -->
                            <el-icon><component :is="menu.children[0].icon" /></el-icon>
                            <template #title>
                                {{menu.children[0].name}}
                            </template>
                        </el-menu-item>
                        <!-- 处理有多个子路由的情况，如集群、工作负载、负载均衡等 -->
                        <!-- 父菜单 -->
                        <!-- 注意el-menu-item在折叠后，title的部分会自动消失，但el-sub-menu不会，需要自己控制 -->
                        <el-sub-menu class="aside-submenu" v-else-if="menu.children && menu.children.length > 1" :index="menu.path">
                            <template #title>
                                <el-icon><component :is="menu.icon" /></el-icon>
                                <span :class="[isCollapse ? 'is-collapse' : '']">{{menu.name}}</span>
                            </template>
                            <!-- 子菜单 -->
                            <el-menu-item class="aside-menu-childitem" v-for="child in menu.children" :key="child" :index="child.path">
                                <template #title>
                                    {{child.name}}
                                </template>
                            </el-menu-item>
                        </el-sub-menu>
                    </div>
                </el-menu>
            </el-aside>
        </el-container>
    </div>
</template>

<script>
import {useRouter} from 'vue-router'
export default {
    data() {
        return {
            //导入logo图片
            logo: require('@/assets/k8s/k8s-metrics.png'),
            //控制导航栏折叠
            isCollapse: false,
            //导航栏宽度
            asideWidth: '220px',
            //路由规则
            routers: [],
        }
    },
    beforeMount() {
        //使用useRouter().options.routes方法获取路由规则
        this.routers = useRouter().options.routes
    }
}
</script>


<style scoped>
    /* 侧边栏折叠速度，背景色 */
    .aside{
        transition: all .5s;
        background-color: #131b27;
    }
    /* 固钉，以及logo图片和平台名的属性 */
    .aside-logo{
        background-color: #131b27;
        height: 60px;
        color: white;
    }
    .logo-image {
        width: 40px;
        height: 40px;
        top: 12px;
        padding-left: 12px;
    }
    .logo-name{
        font-size: 20px;
        font-weight: bold;
        padding: 10px;
    }
    /* 滚动条不展示 */
    .aside::-webkit-scrollbar {
        display: none;
    }
    /* 修整边框，让边框不要有溢出 */
    .aside-affix {
        border-bottom-width: 0;
    }
    .aside-menu {
        border-right-width: 0
    }
    /* 菜单栏的位置以及颜色 */
    .aside-menu-item.is-active {
        background-color: #1f2a3a ;
    }
    .aside-menu-item {
        padding-left: 20px !important;
    }
    .aside-menu-item:hover {
        background-color: #142c4e ;
    }
    .aside-menu-childitem {
        padding-left: 40px !important;
    }
    .aside-menu-childitem.is-active {
        background-color: #1f2a3a ;
    }
    .aside-menu-childitem:hover {
        background-color: #142c4e ;
    }

</style>
```

（4）Header

功能：面包屑、下拉框、登出按钮



```
▼(2) [{…}, {…}]
 ▶0: {path: '/workload', redirect: undefined, name: '工作负载', meta: {…}, aliasOf: undefined, …}
 ▶1: {path: '/workload/deployment', redirect: undefined, name: 'Deployment', meta: {…}, aliasOf: undefined, …}
  length: 2
 ▶[[Prototype]]: Array(0)
```

```
<template>
    <div class="common-layout">
        <!-- container整体布局 -->
        <el-container style="height: 100vh;">
            <!-- header、main、以及footer -->
            <el-container>
                <!-- header -->
                <el-header class="header">
                    <el-row :gutter="20">
                        <el-col :span="1">
                            <!-- 折叠按钮 -->
                            <div class="header-collapse" @click="onCollapse">
                                <el-icon><component :is="isCollapse ? 'expand':'fold'"
/></el-icon>
                            </div>
                        </el-col>
                        <el-col :span="10">
                            <!-- 面包屑 -->
                            <div class="header-breadcrumb">
                                <!-- separator 分隔符 -->
                                <el-breadcrumb separator="/">
                                    <!-- :to="{ path: '/' }"表示跳转到/路径 -->
                                    <el-breadcrumb-item :to="{ path: '/' }">工作台</el-
breadcrumb-item>
                                    <!-- this.$route.matched 可以拿到当前页面的路由信息 -->
                                    <template v-for="(matched,m) in
this.$route.matched" :key="m">
                                        <el-breadcrumb-item v-if="matched.name !=
undefined">
                                            {{ matched.name }}
                                        </el-breadcrumb-item>
                                    </template>
                                </el-breadcrumb>
                            </div>
                        </el-col>
                        <el-col class="header-menu" :span="13">
                            <!-- 用户信息 -->
                            <el-dropdown>
                                <!-- 头像及用户名 -->
                                <div class="header-dropdown">
                                    <el-image class="avator-image" :src="avator" />
                                    <span>{{ username }}</span>
                                </div>
```

```
                                  <!-- 下拉框内容 -->
                                  <template #dropdown>
                                      <el-dropdown-menu>
                                          <el-dropdown-item @click="logout()">退出</el-
dropdown-item>
                                          <el-dropdown-item >修改密码</el-dropdown-item>
                                      </el-dropdown-menu>
                                  </template>
                              </el-dropdown>
                          </el-col>
                      </el-row>
                  </el-header>
              </el-container>
          </el-container>
    </div>
</template>

<script>
export default {
    data() {
        return {
            //导入头像图片
            avator: require('@/assets/avator/avator.png'),
            //控制导航栏折叠
            isCollapse: false,
        }
    },
    computed: {
        //获取用户名
        username() {
            let username = localStorage.getItem('username');
            //三元运算
            return username ? username : '未知';
        },
    },
    methods: {
        //控制折叠
        onCollapse() {
            if (this.isCollapse) {
                this.asideWidth = '220px'
                this.isCollapse = false
            } else {
                this.isCollapse = true
                this.asideWidth = '64px'
            }
        },
        //登出
        logout() {
            //移除用户名
            localStorage.removeItem('username');
            //移除token
            localStorage.removeItem('token');
            //跳转至/login页面
            this.$router.push('/login');
        }
    },
}
</script>
```

```
<style scoped>
    /* header的属性 */
    .header{
        z-index: 1200;
        line-height: 60px;
        font-size: 24px;
        box-shadow: 0 2px 4px rgba(0, 0, 0, .12), 0 0 6px rgba(0, 0, 0, .04)
    }
    /* 折叠按钮 */
    .header-collapse{
        cursor: pointer;
    }
    /* 面包屑 */
    .header-breadcrumb{
        padding-top: 0.9em;
    }
    /* 用户信息靠右 */
    .header-menu{
        text-align: right;
    }
    /* 折叠属性 */
    .is-collapse {
        display: none;
    }
    /* 用户信息下拉框 */
    .header-dropdown {
        line-height: 60px;
        cursor: pointer;
    }
    /* 头像 */
    .avator-image {
        top: 12px;
        width: 40px;
        height: 40px;
        border-radius: 50%;
        margin-right: 8px;
    }

</style>
```

（5）Main

功能：路由占位符

（6）Footer

```
<template>
    <div class="common-layout">
        <!-- container整体布局 -->
        <el-container style="height: 100vh;">
            <!-- header、main、以及footer -->
            <el-container>
                <!-- main -->
                <el-main class="main">
                    <!-- 路由占位符，展示匹配到的路由的视图组件 -->
                    <router-view></router-view>
                </el-main>
                <!-- footer -->
                <el-footer class="footer">
                    <el-icon style="width:2em;top:3px;font-size:18px"><place/></el-
icon>
                    <a class="footer el-icon-place">2022 adoo devops</a>
                </el-footer>
                <!-- 返回顶部，其实是返回el-main的顶部 -->
                <el-backtop target=".el-main"></el-backtop>
            </el-container>
        </el-container>
    </div>
</template>


<style scoped>
    .main {
        padding: 10px;
    }
    .footer {
        z-index: 1200;
        color: rgb(187, 184, 184);
        font-size: 14px;
        text-align: center;
        line-height: 60px;
    }

</style>
```

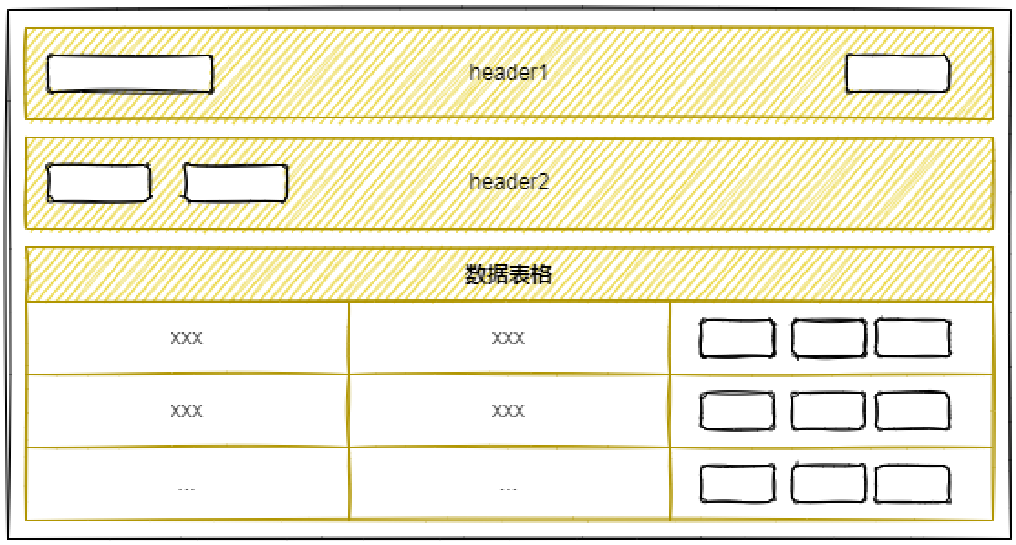
2、工作负载

2.1 Deployment

（1）功能

列表、详情、新增、更新、删除、重启、副本数

（2）Main布局



```
<template>
    <div class="deploy">
        <el-row>
            <!-- 头部1 -->
            <el-col :span="24"></el-col>
            <!-- 头部2 -->
            <el-col :span="24"></el-col>
            <!-- 数据表格 -->
            <el-col :span="24"></el-col>
        </el-row>
    </div>
</template>
```

（2）头部工具栏一

获取Namespace

```
<template>
    <div class="deploy">
        <el-row>
            <!-- 头部1 -->
            <el-col :span="24">
                <div>
                    <!-- 包一层卡片 -->
                    <el-card class="deploy-head-card" shadow="never" :body-style="
{padding:'10px'}">
                        <el-row>
                            <!-- 命名空间的下拉框 -->
                            <el-col :span="6">
                                <div>
                                    <span>命名空间：</span>
                                    <!-- 下拉框 -->
                                    <!-- filterable：带搜索功能 -->
                                    <!-- placeholder 默认提示 -->
                                    <!-- label 显示内容 -->
                                    <!-- value 绑定到v-model的值中 -->
                                    <el-select v-model="namespaceValue" filterable
placeholder="请选择">
                                        <el-option
                                        v-for="(item, index) in namespaceList"
                                        :key="index"
                                        :label="item.metadata.name"
                                        :value="item.metadata.name">
                                        </el-option>
                                    </el-select>
                                </div>
                            </el-col>
                            <!-- 刷新按钮 -->
                            <el-col :span="2" :offset="16">
                                <div>
                                    <!-- 每次刷新，都重新调一次list接口，刷新表格中的数据 -->
                                    <el-button style="border-radius:2px;"
icon="Refresh" plain>刷新</el-button>
                                </div>
                            </el-col>
                        </el-row>
                    </el-card>
                </div>
            </el-col>
        </el-row>
    </div>
</template>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //命名空间
            namespaceValue: 'default',
            namespaceList: [],
            namespaceListUrl: common.k8sNamespaceList,
        }
    },
    methods: {
        //获取Namespace列表
        getNamespaces() {
            httpClient.get(this.namespaceListUrl)
            .then(res => {
                this.namespaceList = res.data.items
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        },
    },
    watch: {
        //监听namespace的值,若发生变化，则执行handler方法中的内容
        namespaceValue: {
            handler() {
                //将namespace的值存入本地，用于path切换时依旧能获取得到
                localStorage.setItem('namespace', this.namespaceValue)
            }
        },
    },
    beforeMount() {
        //加载页面时先获取localStorage中的namespace值，若获取不到则默认default
        if (localStorage.getItem('namespace') !== undefined &&
localStorage.getItem('namespace') !== null) {
            this.namespaceValue = localStorage.getItem('namespace')
        }
        this.getNamespaces()
    }
}
</script>


<style scoped>
    /* 卡片属性 */
    .deploy-head-card,.deploy-body-card {
        border-radius: 1px;
        margin-bottom: 5px;
    }
</style>
```

（3）头部工具栏二

创建Deployment

```
<template>
    <div class="deploy">
        <el-row>
            <!-- 头部2 -->
            <el-col :span="24">
                <div>
                    <!-- 包一层卡片 -->
                    <el-card class="deploy-head-card" shadow="never" :body-style="
{padding:'10px'}">
                        <el-row>
                            <!-- 创建按钮 -->
                            <el-col :span="2">
                                <div>
                                    <!-- 点击后打开抽屉，填入创建deployment需要的数据 -->
                                    <el-button style="border-radius:2px;" icon="Edit"
type="primary" @click="createDeploymentDrawer = true" v-
loading.fullscreen.lock="fullscreenLoading">创建</el-button>
                                </div>
                            </el-col>
                            <!-- 搜索框和搜索按钮 -->
                            <el-col :span="6">
                                <div>
                                    <el-input class="deploy-head-search" clearable
placeholder="请输入" v-model="searchInput"></el-input>
                                    <el-button style="border-radius:2px;"
icon="Search" type="primary" plain>搜索</el-button>
                                </div>
                            </el-col>
                        </el-row>
                    </el-card>
                </div>
            </el-col>
        </el-row>
        <!-- 抽屉：创建Deployment的表单 -->
        <!-- v-model 值是bool，用于显示与隐藏 -->
        <!-- direction 显示的位置 -->
        <!-- before-close 关闭时触发，点击关闭或者点击空白都会触发 -->
        <el-drawer
            v-model="createDeploymentDrawer"
            :direction="direction"
            :before-close="handleClose">
            <!-- 插槽，抽屉标题 -->
            <template #title>
                <h4>创建Deployment</h4>
            </template>
            <!-- 插槽，抽屉body -->
            <template #default>
                <!-- flex布局,居中 -->
                <el-row type="flex" justify="center">
                    <el-col :span="20">
                        <!-- ref绑定控件后，js中才能用this.$ref获取该控件 -->
                        <!-- rules 定义form表单校验规则 -->
                        <el-form ref="createDeployment" :rules="createDeploymentRules"
:model="createDeployment" label-width="80px">
                            <!-- prop用于rules中的校验规则的key -->
                            <el-form-item class="deploy-create-form" label="名称"
prop="name">
                                <el-input v-model="createDeployment.name"></el-input>
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="命名空间"
prop="namespace">
                                <el-select v-model="createDeployment.namespace"
filterable placeholder="请选择">
                                    <el-option
                                    v-for="(item, index) in namespaceList"
                                    :key="index"
                                    :label="item.metadata.name"
                                    :value="item.metadata.name">
                                    </el-option>
                                </el-select>
                            </el-form-item>
                            <!-- 数字输入框，最小为1，最大为10 -->
                            <el-form-item class="deploy-create-form" label="副本数"
prop="replicas">
                                <el-input-number v-model="createDeployment.replicas"
:min="1" :max="10"></el-input-number>
                                    <!-- 气泡弹出框用于提醒上限 -->
                                    <el-popover
                                        placement="top"
                                        :width="100"
                                        trigger="hover"
                                        content="申请副本数上限为10个">
                                        <template #reference>
                                            <el-icon style="width:2em;font-
size:18px;color:#4795EE"><WarningFilled/></el-icon>
                                        </template>
                                    </el-popover>
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="镜像"
prop="image">
                                <el-input v-model="createDeployment.image"></el-input>
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="标签"
prop="label_str">
                                <el-input v-model="createDeployment.label_str"
placeholder="示例: project=ms,app=gateway"></el-input>
                            </el-form-item>
                            <!-- 下拉框，用于规格的选择，之后用/分割，得到cpu和内存 -->
                            <el-form-item class="deploy-create-form" label="资源配额"
prop="resource">
                                <el-select v-model="createDeployment.resource"
placeholder="请选择">
                                    <el-option value="0.5/1" label="0.5C1G"></el-
option>
                                    <el-option value="1/2" label="1C2G"></el-option>
                                    <el-option value="2/4" label="2C4G"></el-option>
                                    <el-option value="4/8" label="4C8G"></el-option>
                                </el-select>
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="容器端口"
prop="container_port">
                                <el-input v-model="createDeployment.container_port"
placeholder="示例: 80"></el-input>
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="健康检查"
prop="health">
                                <el-switch v-model="createDeployment.health_check" />
                            </el-form-item>
                            <el-form-item class="deploy-create-form" label="检查路径"
prop="healthPath">
                                <el-input v-model="createDeployment.health_path"
placeholder="示例: /health"></el-input>
                            </el-form-item>
                        </el-form>
                    </el-col>
                </el-row>
            </template>
            <!-- 插槽，抽屉footer -->
            <template #footer>
                <!-- 点击后赋值false，隐藏抽屉 -->
                <el-button @click="createDeploymentDrawer = false">取消</el-button>
                <el-button type="primary" @click="submitForm('createDeployment')">立即
创建</el-button>
            </template>
        </el-drawer>
    </div>
</template>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //搜索框内容
            searchInput: '',
            //创建
            fullscreenLoading: false,
            direction: 'rtl',
            createDeploymentDrawer: false,
            createDeployment: {
                name: '',
                namespace: '',
                replicas: 1,
                image: '',
                resource: '',
                health_check: false,
                health_path: '',
                label_str: '',
                label: {},
                container_port: ''
            },
            //创建请求的参数
            createDeploymentData: {
                url: common.k8sDeploymentCreate,
                params: {}
            },
            //创建deployment的表单校验规则
            createDeploymentRules: {
                name: [{
                    required: true,
                    message: '请填写名称',
                    trigger: 'change'
                }],
                image: [{
                    required: true,
                    message: '请填写镜像',
                    trigger: 'change'
                }],
                namespace: [{
                    required: true,
                    message: '请选择命名空间',
                    trigger: 'change'
                }],
                resource: [{
                    required: true,
                    message: '请选择配额',
                    trigger: 'change'
                }],
                label_str: [{
                    required: true,
                    message: '请填写标签',
                    trigger: 'change'
                }],
                container_port: [{
                    required: true,
                    message: '请填写容器端口',
                    trigger: 'change'
                }],
            },
        }
    },
    methods: {
        //处理抽屉的关闭，增加体验感
        handleClose(done) {
            this.$confirm('确认关闭？')
            .then(() => {
                done();
            })
            .catch(() => {});
        },
        //创建deployment，加Func的原因是因为createDeploy用于属性了
        createDeployFunc() {
            //正则匹配，验证label的合法性
            let reg = new RegExp("(^[A-Za-z]+=[A-Za-z0-9]+).*")
            if (!reg.test(this.createDeployment.label_str)) {
                this.$message.warning({
                    message: "标签填写异常，请确认后重新填写"
                })
                return
            }
            //加载loading动画
            this.fullscreenLoading = true
            //定义label、cpu和memory变量
            //'app=xxx,version=yyy'
            //['app=xxx','version=yyy']
            //[['app','xxx'],['version','yyy']]
            //map['app']='xxx',map['version']=yyy
            let label = new Map()
            let cpu, memory
            //将label字符串转成数组
            let a = (this.createDeployment.label_str).split(",")
            //将数组转成map
            a.forEach(item => {
                let b = item.split("=")
                label[b[0]] = b[1]
            })
            //将deployment的规格转成cpu和memory
            let resourceList = this.createDeployment.resource.split("/")
            cpu = resourceList[0]
            memory = resourceList[1] + "Gi"
            //赋值
            this.createDeploymentData.params = this.createDeployment
            this.createDeploymentData.params.container_port =
parseInt(this.createDeployment.container_port)
            this.createDeploymentData.params.label = label
            this.createDeploymentData.params.cpu = cpu
            this.createDeploymentData.params.memory = memory
            httpClient.post(this.createDeploymentData.url,
this.createDeploymentData.params)
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                //创建后重新获取列表
                this.getDeployments()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
```

```
                //重置表单
                this.resetForm('createDeployment')
                //关闭加载动画
                this.fullscreenLoading = false
                //关闭抽屉
                this.createDeploymentDrawer = false
            },
            //重置表单方法，element plus课程讲过的
            resetForm(formName) {
                this.$refs[formName].resetFields()
            },
            //提交表单，校验参数合法性
            submitForm(formName) {
                this.$refs[formName].validate((valid) => {
                    if (valid) {
                        this.createDeployFunc()
                    } else {
                        return false;
                    }
                })
            }
        }
    }
</script>


<style scoped>
    /* 卡片属性 */
    .deploy-head-card,.deploy-body-card {
        border-radius: 1px;
        margin-bottom: 5px;
    }
    /* 搜索框 */
    .deploy-head-search {
        width:160px;
        margin-right:10px;
    }
</style>
```

（4）引入codemirror编辑器

main.js

```
//codemirror编辑器
import { GlobalCmComponent } from "codemirror-editor-vue3";
// 引入主题 可以从 codemirror/theme/ 下引入多个
import 'codemirror/theme/idea.css'
// 引入语言模式 可以从 codemirror/mode/ 下引入多个
import 'codemirror/mode/yaml/yaml.js'
```

src/views/common/Config.js

```
    //编辑器配置
    cmOptions: {
        // 语言及语法模式
        mode: 'text/yaml',
        // 主题
        theme: 'idea',
        // 显示行数
        lineNumbers: true,
        smartIndent: true, //智能缩进
        indentUnit: 4, // 智能缩进单元长度为 4 个空格
        styleActiveLine: true, // 显示选中行的样式
        matchBrackets: true, //每当光标位于匹配的方括号旁边时，都会使其高亮显示
        readOnly: false,
        lineWrapping: true //自动换行
    }
```

（4）数据表格

列表

```
<template>
    <div class="deploy">
        <el-row>
            <!-- 数据表格 -->
            <el-col :span="24">
                <div>
                    <!-- 包一层卡片 -->
                    <el-card class="deploy-body-card" shadow="never" :body-style="
{padding:'5px'}">
                        <!-- 数据表格 -->
                        <!-- v-loading用于加载时的loading动画 -->
                        <el-table
                        style="width:100%;font-size:12px;margin-bottom:10px;"
                        :data="deploymentList"
                        v-loading="appLoading">
                            <!-- 最左侧留出20px的宽度，更加没关 -->
                            <el-table-column width="20"></el-table-column>
                            <!-- deployment名字 -->
                            <el-table-column align=left label="Deployment名">
                                <!-- 插槽，scope.row获取当前行的数据 -->
                                <template v-slot="scope">
                                    <a class="deploy-body-deployname">{{
scope.row.metadata.name }}</a>
                                </template>
                            </el-table-column>
                            <!-- 标签 -->
                            <el-table-column align=center label="标签">
                                <template v-slot="scope">
                                    <!-- for循环，每个label只显示固定长度，鼠标悬停后气泡弹出
框显示完整长度 -->
                                    <div v-for="(val, key) in
scope.row.metadata.labels" :key="key">
                                        <!-- 气泡弹出框 -->
                                        <!-- placement 弹出位置 -->
                                        <!-- trigger 触发条件 -->
                                        <!-- content 弹出框内容 -->
                                        <el-popover
                                            placement="right"
                                            :width="200"
                                            trigger="hover"
                                            :content="key + ':' + val">
                                            <template #reference>
                                                <!-- ellipsis方法用于剪裁字符串 -->
                                                <el-tag style="margin-bottom: 5px"
type="warning">{{ ellipsis(key + ":" + val) }}</el-tag>
                                            </template>
                                        </el-popover>
                                    </div>
                                </template>
                            </el-table-column>
                            <!-- 容器组 -->
                            <el-table-column align=center label="容器组">
                                <!-- 可用数量/总数量,三元运算，若值大于0则显示值，否则显示0 -->
                                <template v-slot="scope">
                                    <span>{{ scope.row.status.availableReplicas>0?
scope.row.status.availableReplicas:0  }} / {{ scope.row.spec.replicas>0?
scope.row.spec.replicas:0 }} </span>
                                </template>
                            </el-table-column>
                            <!-- 创建时间 -->
                            <el-table-column align=center min-width="100" label="创建时
间">
                                <!-- timeTrans函数用于将格林威治时间转成北京时间 -->
                                <template v-slot="scope">
                                    <el-tag type="info">{{
timeTrans(scope.row.metadata.creationTimestamp) }} </el-tag>
                                </template>
                            </el-table-column>
                            <!-- 容器镜像 -->
                            <el-table-column align=center label="镜像">
                                <!-- 与label的显示逻辑一致 -->
                                <template v-slot="scope">
                                    <div v-for="(val, key) in
scope.row.spec.template.spec.containers" :key="key">
                                        <el-popover
                                            placement="right"
                                            :width="200"
                                            trigger="hover"
                                            :content="val.image">
                                            <template #reference>
                                                <el-tag style="margin-bottom: 5px">{{
ellipsis(val.image.split('/')[2]==undefined?val.image:val.image.split('/')[2]) }}</el-
tag>
                                            </template>
                                        </el-popover>
                                    </div>
                                </template>
                            </el-table-column>
                            <!-- 操作列，放按钮 -->
                            <el-table-column align=center label="操作" width="400">
                                <template v-slot="scope">
                                    <el-button size="small" style="border-radius:2px;"
icon="Edit" type="primary" plain @click="getDeploymentDetail(scope)">YAML</el-button>
                                    <el-button size="small" style="border-radius:2px;"
icon="Plus" type="primary" @click="handleScale(scope)">扩缩</el-button>
                                    <el-button size="small" style="border-radius:2px;"
icon="RefreshLeft" type="primary" @click="handleConfirm(scope, '重启',
restartDeployment)">重启</el-button>
                                    <el-button size="small" style="border-radius:2px;"
icon="Delete" type="danger" @click="handleConfirm(scope, '删除', delDeployment)">删除
</el-button>
```

```
                                </template>
                            </el-table-column>
                        </el-table>
                        <!-- 分页配置 -->
                        <!-- background 背景色灰 -->
                        <!-- size-change 单页大小改变后触发 -->
                        <!-- current-change 页数改变后触发 -->
                        <!-- current-page 当前页 -->
                        <!-- page-size 单页大小 -->
                        <!-- layout 分页器支持的功能 -->
                        <!-- total 数据总条数 -->
                        <el-pagination
                        class="deploy-body-pagination"
                        background
                        @size-change="handleSizeChange"
                        @current-change="handleCurrentChange"
                        :current-page="currentPage"
                        :page-sizes="pagesizeList"
                        :page-size="pagesize"
                        layout="total, sizes, prev, pager, next, jumper"
                        :total="deploymentTotal">
                        </el-pagination>
                    </el-card>
                </div>
            </el-col>
        </el-row>
    </div>
</template>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //分页
            currentPage: 1,
            pagesize: 10,
            pagesizeList: [10, 20, 30],
            //列表
            appLoading: false,
            deploymentList: [],
            deploymentTotal: 0,
            getDeploymentsData: {
                url: common.k8sDeploymentList,
                params: {
                    filter_name: '',
                    namespace: '',
                    page: '',
                    limit: '',
                }
            }
        }
    },
    methods: {
        //页面大小发生变化时触发，赋值并重新获取列表
        handleSizeChange(size) {
            this.pagesize = size;
            this.getDeployments()
        },
        //页数发生变化时触发，复制并重新获取列表
        handleCurrentChange(currentPage) {
            this.currentPage = currentPage;
            this.getDeployments()
        },
        //字符串截取、拼接并返回
        ellipsis(value) {
            return value.length>15?value.substring(0,15)+'...':value
        },
        //格林威治时间转为北京时间
        timeTrans(timestamp) {
            let date = new Date(new Date(timestamp).getTime() + 8 * 3600 * 1000)
            date = date.toJSON();
            date = date.substring(0, 19).replace('T', ' ')
            return date
        },
        //获取Deployment列表
        getDeployments() {
            //表格加载动画开启
            this.appLoading = true
            //getDeploymentsData是用于发起deployment列表请求的专用的对象，里面有url和params参
数,以下是赋值
            this.getDeploymentsData.params.filter_name = this.searchInput
            this.getDeploymentsData.params.namespace = this.namespaceValue
            this.getDeploymentsData.params.page = this.currentPage
            this.getDeploymentsData.params.limit = this.pagesize
            httpClient.get(this.getDeploymentsData.url, {params:
this.getDeploymentsData.params})
            .then(res => {
                //响应成功，获取deployment列表和total
                this.deploymentList = res.data.items
                this.deploymentTotal = res.data.total
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
            //加载动画关闭
            this.appLoading = false
        }
    },
    watch: {
        //监听namespace的值,若发生变化，则执行handler方法中的内容
        namespaceValue: {
            handler() {
                //将namespace的值存入本地，用于path切换时依旧能获取得到
                localStorage.setItem('namespace', this.namespaceValue)
                //重置当前页为1
                this.currentPage = 1
                //获取deployment列表
                this.getDeployments()
            }
        },
    },
    beforeMount() {
        //加载页面时先获取localStorage中的namespace值，若获取不到则默认default
        if (localStorage.getItem('namespace') !== undefined &&
localStorage.getItem('namespace') !== null) {
            this.namespaceValue = localStorage.getItem('namespace')
        }
        //this.getNamespaces()
        this.getDeployments()
    }
}
</script>


<style scoped>
    /* 数据表格deployment名颜色 */
    .deploy-body-deployname {
        color: #4795EE;
    }
    /* deployment名鼠标悬停 */
    .deploy-body-deployname:hover {
        color: rgb(84, 138, 238);
        cursor: pointer;
        font-weight: bold;
    }
</style>
```

详情（YAML）

```
<el-table-column align=center label="操作" width="400">
        <template v-slot="scope">
                <el-button size="small" style="border-radius:2px;" icon="Edit"
type="primary" plain @click="getDeploymentDetail(scope)">YAML</el-button>
        </template>
</el-table-column>
        <!-- 展示YAML信息的弹框 -->
        <el-dialog title="YAML信息" v-model="yamlDialog" width="45%" top="2%">
            <!-- codemirror编辑器 -->
            <!-- border 带边框 -->
            <!-- options  编辑器配置 -->
            <!-- change 编辑器中的内容变化时触发 -->
            <codemirror
                :value="contentYaml"
                border
                :options="cmOptions"
                height="500"
                style="font-size:14px;"
                @change="onChange"
            ></codemirror>
            <template #footer>
                <span class="dialog-footer">
                    <el-button @click="yamlDialog = false">取 消</el-button>
                    <el-button type="primary" @click="updateDeployment()">更 新</el-
button>
                </span>
            </template>
        </el-dialog>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
import yaml2obj from 'js-yaml';
import json2yaml from 'json2yaml';
export default {
    data() {
        return {
            //编辑器配置
            cmOptions: common.cmOptions,
            contentYaml: '',
            //详情
            deploymentDetail: {},
            getDeploymentDetailData: {
                url: common.k8sDeploymentDetail,
                params: {
                    deployment_name: '',
                    namespace: ''
                }
            },
            //yaml更新
            yamlDialog: false,
            updateDeploymentData: {
                url: common.k8sDeploymentUpdate,
                params: {
                    namespace: '',
                    content: ''
                }
            }
        }
    },
    methods: {
        //json转yaml方法
        transYaml(content) {
            return json2yaml.stringify(content)
        },
        //yaml转对象
        transObj(content) {
            return yaml2obj.load(content)
        },
        //编辑器内容变化时触发的方式,用于将更新的内容复制到变量中
        onChange(val) {
            this.contentYaml = val
        },
        //获取deployment详情，e参数标识传入的scope插槽，.row是该行的数据
        getDeploymentDetail(e) {
            this.getDeploymentDetailData.params.deployment_name = e.row.metadata.name
            this.getDeploymentDetailData.params.namespace = this.namespaceValue
            httpClient.get(this.getDeploymentDetailData.url, {params:
this.getDeploymentDetailData.params})
            .then(res => {
                //响应成功，获得deployment详情
                this.deploymentDetail = res.data
                //将对象转成yaml格式的字符串
                this.contentYaml = this.transYaml(this.deploymentDetail)
                //打开弹出框
                this.yamlDialog = true
            })
            .catch(res => {
                this.$message.error({
```

```
                message: res.msg
                })
            })
        },
        //更新deployment
        updateDeployment() {
            //将yaml格式的deployment对象转为json
            let content = JSON.stringify(this.transObj(this.contentYaml))
            this.updateDeploymentData.params.namespace = this.namespaceValue
            this.updateDeploymentData.params.content = content
            httpClient.put(this.updateDeploymentData.url,
this.updateDeploymentData.params)
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                //更新后重新获取列表
                this.getDeployments()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
            //关闭弹出框
            this.yamlDialog = false
        }
    }
}
</script>
```

伸缩

```
<el-table-column align=center label="操作" width="400">
        <template v-slot="scope">
                <el-button size="small" style="border-radius:2px;" icon="Plus"
type="primary" @click="handleScale(scope)">扩缩</el-button>
        </template>
</el-table-column>
        <!-- 调整副本数的弹框 -->
        <el-dialog title="副本数调整" v-model="scaleDialog" width="25%">
            <div style="text-align:center">
                <span>实例数: </span>
                <el-input-number :step="1" v-model="scaleNum" :min="0" :max="30"
label="描述文字"></el-input-number>
            </div>
            <template #footer>
                <span class="dialog-footer">
                    <el-button @click="scaleDialog = false">取 消</el-button>
                    <el-button type="primary" @click="scaleDeployment()">更 新</el-
button>
                </span>
            </template>
        </el-dialog>


<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
```

```
export default {
    data() {
        return {
            //扩缩容
            scaleNum: 0,
            scaleDialog: false,
            scaleDeploymentData: {
                url: common.k8sDeploymentScale,
                params: {
                    deployment_name: '',
                    namespace: '',
                    scale_num: ''
                }
            }
        }
    },
    methods: {
        //扩缩容的中间方法，用于赋值及打开弹出框
        handleScale(e) {
            this.scaleDialog = true
            this.deploymentDetail = e.row
            this.scaleNum = e.row.spec.replicas
        },
        //扩缩容deployment
        scaleDeployment() {
            this.scaleDeploymentData.params.deployment_name =
this.deploymentDetail.metadata.name
            this.scaleDeploymentData.params.namespace = this.namespaceValue
            this.scaleDeploymentData.params.scale_num = this.scaleNum
            httpClient.put(this.scaleDeploymentData.url,
this.scaleDeploymentData.params)
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                //更新后重新获取列表
                this.getDeployments()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
            //关闭弹出框
            this.scaleDialog = false
        }
    }
}
</script>
```

重启

```
<el-table-column align=center label="操作" width="400">
        <template v-slot="scope">
                <el-button size="small" style="border-radius:2px;" icon="RefreshLeft"
type="primary" @click="handleConfirm(scope, '重启', restartDeployment)">重启</el-button>
```

```
        </template>
</el-table-column>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //重启
            restartDeploymentData: {
                url: common.k8sDeploymentRestart,
                params: {
                    deployment_name: '',
                    namespace: '',
                }
            }
        }
    },
    methods: {
        //重启deployment
        restartDeployment(e) {
            this.restartDeploymentData.params.deployment_name = e.row.metadata.name
            this.restartDeploymentData.params.namespace = this.namespaceValue
            httpClient.put(this.restartDeploymentData.url,
this.restartDeploymentData.params)
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                this.getDeployments()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        },
        //弹出确认框，用于危险操作的double check
        //obj是行数据，opeateName是操作名，fn是操作的方法
        handleConfirm(obj, operateName, fn) {
            this.confirmContent = '确认继续 ' + operateName + ' 操作吗？'
            //$confirm用于弹出确认框
            this.$confirm(this.confirmContent,'提示',{
                confirmButtonText: '确定',
                cancelButtonText: '取消',
            })
            .then(() => {
                //restartDeployment(e)或者delDeployment(e)
                fn(obj)
            })
            .catch(() => {
                this.$message.info({
                    message: '已取消操作'
                })
            })
        }
    }
}
</script>
```

删除

```
<el-table-column align=center label="操作" width="400">
        <template v-slot="scope">
                <el-button size="small" style="border-radius:2px;" icon="Delete"
type="danger" @click="handleConfirm(scope, '删除', delDeployment)">删除</el-button>
        </template>
</el-table-column>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //删除
            delDeploymentData: {
                url: common.k8sDeploymentDel,
                params: {
                    deployment_name: '',
                    namespace: '',
                }
            }
        }
    },
    methods: {
        //删除deployment
        delDeployment(e) {
            this.delDeploymentData.params.deployment_name = e.row.metadata.name
            this.delDeploymentData.params.namespace = this.namespaceValue
            httpClient.delete(this.delDeploymentData.url, {data:
this.delDeploymentData.params})
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                this.getDeployments()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        }
    }
}
</script>
```

2.2 Pod

（1）功能

列表、详情、更新、删除、日志、终端

（2）布局

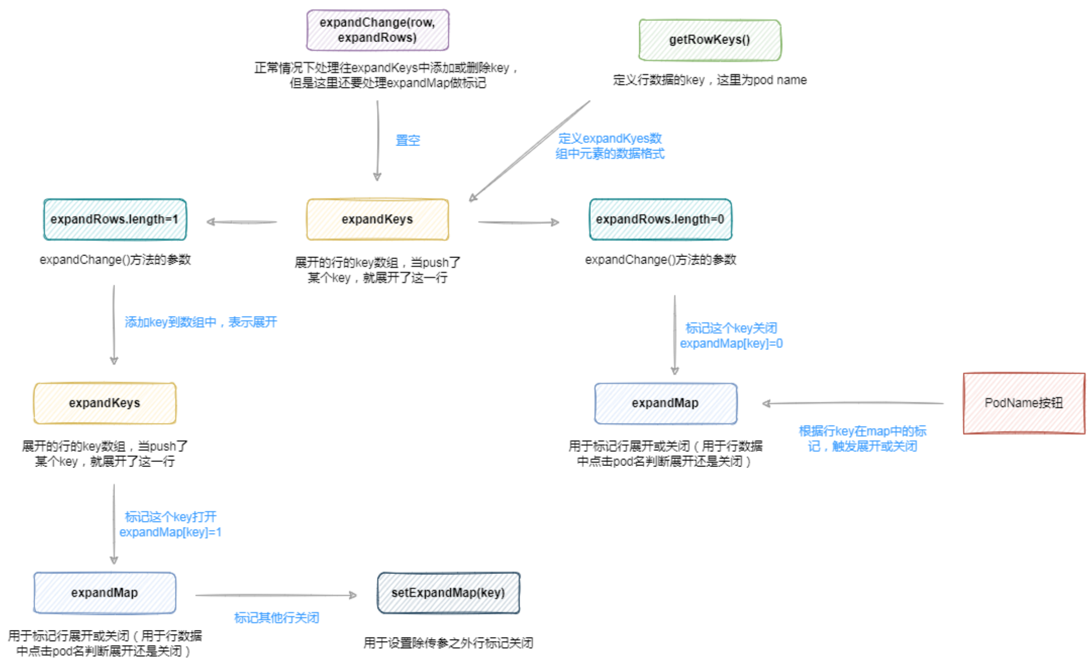
（3）头部工具栏

（4）数据表格

pod信息

```
<el-table-column align=left label="Pod名">
    <template v-slot="scope">
        <a class="pod-body-podname" >{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center min-width="150" label="节点">
    <template v-slot="scope">
        <el-tag v-if="scope.row.spec.nodeName !== undefined" type="warning">{{
scope.row.spec.nodeName }}</el-tag>
    </template>
</el-table-column>
<el-table-column align=center label="状态">
    <template v-slot="scope">
        <div :class="{'success-dot':scope.row.status.phase == 'Running', 'warning-
dot':scope.row.status.phase == 'Pending', 'error-dot':scope.row.status.phase !=
'Running' && scope.row.status.phase != 'Pending'}"></div>
        <span :class="{'success-status':scope.row.status.phase == 'Running', 'warning-
status':scope.row.status.phase ==     'Pending', 'error-status':scope.row.status.phase
!= 'Running' && scope.row.status.phase != 'Pending'}">{{ scope.row.status.phase }}
</span>
    </template>
</el-table-column>
<el-table-column align=center label="重启数">
    <template v-slot="scope">
        <span>{{ restartTotal(scope) }} </span>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
<el-table-column align=center label="操作" width="200">
    <template v-slot="scope">
        <el-button size="small" style="border-radius:2px;" icon="Edit" type="primary"
plain @click="getPodDetail(scope)">YAML</el-button>
        <el-button size="small" style="border-radius:2px;" icon="Delete" type="danger"
@click="handleConfirm(scope, '删除', delPod)">删除</el-button>
    </template>
</el-table-column>

<script>
export default() {
    methods: {
        restartTotal(e) {
            let index, sum = 0
            let containerStatuses = e.row.status.containerStatuses
            for ( index in containerStatuses) {
                sum = sum + containerStatuses[index].restartCount
            }
            return sum
        }
    }
}
</script>

<style scoped>
    /* pod状态栏圆点的css实现 */
    .success-dot{
        display:inline-block;
        width: 7px;
        height:7px;
        background: rgb(27, 202, 21);
        border-radius:50%;
        border:1px solid rgb(27, 202, 21);
        margin-right: 10px;
    }
    .warning-dot{
        display:inline-block;
        width: 7px;
        height:7px;
        background: rgb(233, 200, 16);
        border-radius:50%;
        border:1px solid rgb(233, 200, 16);
        margin-right: 10px;
    }
    .error-dot{
        display:inline-block;
        width: 7px;
        height:7px;
        background: rgb(226, 23, 23);
        border-radius:50%;
        border:1px solid rgb(226, 23, 23);
        margin-right: 10px;
    }
    .success-status {
        color: rgb(27, 202, 21);
    }
    .warning-status {
        color: rgb(233, 200, 16);
    }
    .error-status {
        color: rgb(226, 23, 23);
    }
</style>
```

展开Expand


expandChange(row, expandRows)
正常情况下处理往expandKeys中添加或删除key，但是这里还要处理expandMap做标记
getRowKeys()
定义行数据的key，这里为pod name
置空
定义expandKyes数组中元素的数据格式
expandRows.length=1
expandChange()方法的参数
expandKeys
展开的行的key数组，当push了某个key，就展开了这一行
expandRows.length=0
expandChange()方法的参数
添加key到数组中，表示展开
标记这个key关闭 expandMap[key]=0
expandKeys
展开的行的key数组，当push了某个key，就展开了这一行
expandMap
用于标记行展开或关闭（用于行数据中点击pod名判断展开还是关闭）
根据行key在map中的标记，触发展开或关闭
PodName按钮
标记这个key打开 expandMap[key]=1
expandMap
用于标记行展开或关闭（用于行数据中点击pod名判断展开还是关闭）
标记其他行关闭
setExpandMap(key)
用于设置除传参之外行标记关闭

```
<!-- 数据表格 -->
<!-- row-key 用来定义行数据的key，结合expand-row-keys使用，往expandKeys中增加key来展开行 -->
<!-- expand-row-keys 展开的行的key数组 -->
<!-- expand-change 展开触发时，调用这个方法 -->
<el-table
style="width:100%;font-size:12px;margin-bottom:10px;"
:data="podList"
v-loading="appLoading"
:row-key="getRowKeys"
:expand-row-keys="expandKeys"
@expand-change="expandChange">
    <el-table-column width="10"></el-table-column>
    <!-- 展开 -->
    <el-table-column type="expand">
        <!-- 插槽，里面是展开的内容,props标识展开的行的数据 -->
        <template #default="props">
            <el-tabs v-model="activeName" type="card">
                <!-- tab容器标签页 -->
                <el-tab-pane label="容器" name="container"></el-tab-pane>
                <!-- tab日志标签页 -->
                <el-tab-pane label="日志" name="log"></el-tab-pane>
                <!-- tab终端标签页 -->
                <el-tab-pane label="终端" name="shell"></el-tab-pane>
            </el-tabs>
    </el-table-column>
    <el-table-column align=left label="Pod名">
        <template v-slot="scope">
            <!-- 三元运算：expandMap[scope.row.metadata.name]为1则
            触发关闭（expandedRows为空数组），为0则触发展开expandedRows有值 -->
            <a class="pod-body-podname" @click="expandMap[scope.row.metadata.name] ?
expandChange(scope.row, []) : expandChange(scope.row, [scope.row])">{{
scope.row.metadata.name }}</a>
        </template>
    </el-table-column>
</el-table>
<script>
```

```
export default {
    data() {
        return {
            //expand扩展
            activeName: 'container',
            expandKeys: [],
            expandMap: {},
        }
    },
    methods: {
        getRowKeys(row) {
            return row.metadata.name
        },
        //row，展开的当前行的数据
        //expandedRows，展开的所有行的数据组成的数组，但是这里用法是只会有一行，也就是数组长度永远
为1
        expandChange(row, expandedRows) {
            //初始化变量
            //清空expandKeys，代表关闭所有展开的行
            this.expandKeys = []
            //清空日志内容
            this.logContent= ''
            //清空containervalue，展开时不显示上次的值
            this.containerValue = ''
            //将tab标签页顶部页面调成容器
            this.activeName = 'container'
            //expandedRows.length == 1表示展开，expandedRows.length == 0 表示关闭
            if (expandedRows.length > 0) {
                //expandMap key表示展开过的行的key，值为1表示展开标记，值为0表示关闭标记
                //expandMap用于数据表格点击name的展开，用于判断这一行是展开还是关闭的行为
                this.expandMap[row.metadata.name] = 1
                //将expandMap除了row.metadata.name，其他key的值都置为0
                this.setExpandMap(row.metadata.name)
                //这里才是真正的展开，将row.metadata.name添加到expandKeys数组中展开，然后执行方
法获取container
                row ? (this.expandKeys.push(row.metadata.name),
this.getPodContainer(row)) : ''
            } else {
                //关闭标记
                this.expandMap[row.metadata.name] = 0
            }
        },
        //匹配expandMap中podName，不相等的全都置为0，意为除了podName这行，其他全都标记关闭
        setExpandMap(podName) {
            let key
            for ( key in this.expandMap ) {
                key !== podName ? this.expandMap[key] = 0 : ''
            }
        }
    }
}
</script>

<style scoped>
    /deep/ .el-tabs__item {
        font-size: 12px;
    }
    /deep/ .el-tabs__header {
        margin-bottom: 8px;
    }
</style>
```

容器

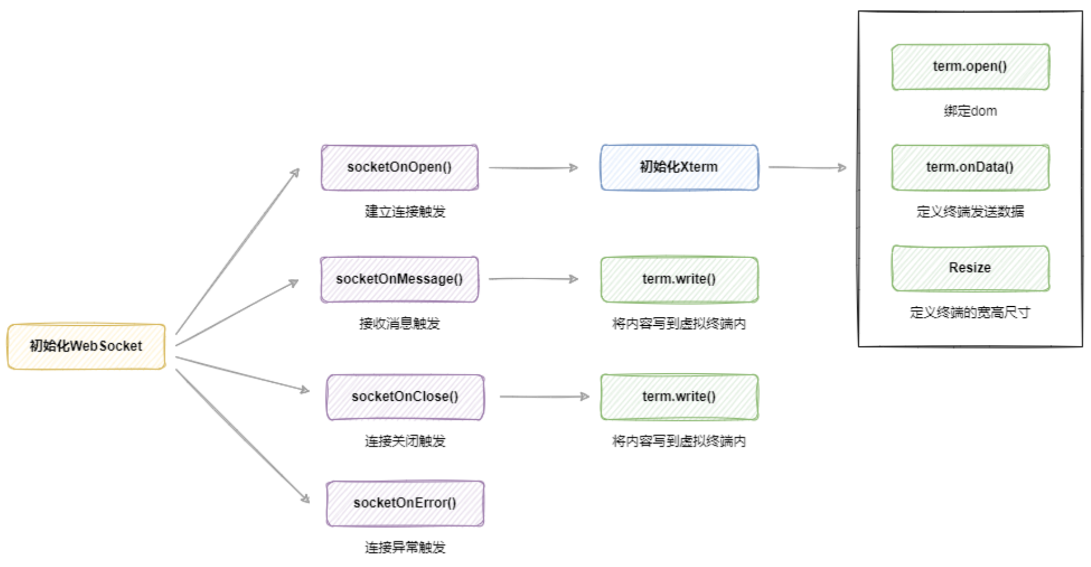
```
<!-- tab容器标签页 -->
<el-tab-pane label="容器" name="container">
    <el-card shadow="never" style="border-radius:1px;" :body-style="{padding:'5px'}">
        <!-- 嵌套数据表格 -->
        <el-table
        style="width:100%;font-size:12px;"
        :data="props.row.spec.containers">
            <el-table-column align=left prop="name" label="容器名"></el-table-column>
            <el-table-column align=left prop="image" label="镜像"></el-table-column>
            <el-table-column align=center label="Pod IP">
                <span>{{ props.row.status.podIP }}</span>
            </el-table-column>
            <el-table-column align=center prop="args" label="启动命令"></el-table-
column>
            <el-table-column align=center label="环境变量">
                <template v-slot="scope">
                    <!-- 气泡弹出框，内容是所有的环境变量 -->
                    <el-popover :width="500" placement="left" trigger="hover">
                        <el-table style="width:100%;font-size:12px;" size="mini"
:show-header="false" :data="scope.row.env">
                            <el-table-column property="name" label="名称"></el-table-
column>
                            <el-table-column property="value" label="值"></el-table-
column>
                        </el-table>
                        <template #reference>
                        <el-button size="small">此处查看</el-button>
                        </template>
                    </el-popover>
                </template>
            </el-table-column>
        </el-table>
    </el-card>
</el-tab-pane>
```

日志

```
<!-- tab日志标签页 -->
<el-tab-pane label="日志" name="log">
    <el-card shadow="never" style="border-radius:1px;" :body-style="{padding:'5px'}">
        <el-row :gutter="10">
            <el-col :span="3">
                <!-- 容器选择框 -->
                <el-select size="small" v-model="containerValue" placeholder="请选择">
                    <el-option v-for="item in containerList" :key="item"
:value="item">
                    </el-option>
                </el-select>
            </el-col>
            <el-col :span="2">
```

```
                  <!-- 查看日志按钮 -->
                  <el-button style="border-radius:2px;" size="small" type="primary"
@click="getPodLog(props.row.metadata.name)">查看</el-button>
              </el-col>
              <el-col :span="24" style="margin-top: 5px">
                  <!-- 显示日志内容 -->
                  <el-card shadow="never" class="pod-body-log-card" :body-style="
{padding:'5px'}">
                      <span class="pod-body-log-span">{{ logContent }}</span>
                  </el-card>
              </el-col>
              </el-row>
      </el-card>
</el-tab-pane>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //日志
            containerList: {},
            containerValue: '',
            getPodContainerData: {
                url: common.k8sPodContainer,
                params: {
                    pod_name: '',
                    namespace: ''
                }
            },
            logContent: '',
            getPodLogData: {
                url: common.k8sPodLog,
                params: {
                    container_name: '',
                    pod_name: '',
                    namespace: ''
                }
            }
        }
    },
    methods: {
        getPodContainer(row) {
            this.getPodContainerData.params.pod_name = row.metadata.name
            this.getPodContainerData.params.namespace = this.namespaceValue
            httpClient.get(this.getPodContainerData.url, {params:
this.getPodContainerData.params})
            .then(res => {
                this.containerList = res.data
                this.containerValue = this.containerList[0]
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        },
        getPodLog(podName) {
            this.getPodLogData.params.pod_name = podName
            this.getPodLogData.params.container_name = this.containerValue
            this.getPodLogData.params.namespace = this.namespaceValue
            httpClient.get(this.getPodLogData.url, {params: 
this.getPodLogData.params})
            .then(res => {
                this.logContent = res.data
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        }
    },
    watch: {
        //若tab标签页切到日志，则重新加载日志内容
        activeName: {
            handler() {
                if ( this.activeName == 'log' ) {
                    this.expandKeys.length == 1 ? this.getPodLog(this.expandKeys[0]) : 
''
                }
            }
        }
    }
}
</script>

<style scoped>
    .pod-body-log-card, .pod-body-shell-card {
        border-radius:1px;
        height:600px;
        overflow:auto;
        background-color: #060101;
    }
    .pod-body-log-card {
        color: aliceblue;
    }
    .pod-body-log-span {
        white-space:pre;
    }
</style>
```

webshell终端



```
<!-- tab终端标签页 -->
<el-tab-pane label="终端" name="shell">
    <el-card shadow="never" style="border-radius:1px;" :body-style="{padding:'5px'}">
        <el-row :gutter="10">
            <el-col :span="3">
                <!-- 容器选择框 -->
                <el-select size="small" v-model="containerValue" placeholder="请选择">
                    <el-option v-for="item in containerList" :key="item"
:value="item">
                    </el-option>
                </el-select>
            </el-col>
            <el-col :span="1">
                <!-- 连接按钮 -->
                <el-button style="border-radius:2px;" size="small" type="primary"
@click="initSocket(props.row)">连接</el-button>
            </el-col>
            <el-col :span="1">
                <!-- 关闭连接按钮 -->
                <el-button style="border-radius:2px;" size="small" type="danger"
@click="closeSocket()">关闭</el-button>
            </el-col>
            <el-col :span="24" style="margin-top: 5px">
                <el-card shadow="never" class="pod-body-shell-card" :body-style="
{padding:'5px'}">
                    <!-- xterm虚拟终端 -->
                    <div id="xterm"></div>
                </el-card>
            </el-col>
            </el-row>
    </el-card>
</el-tab-pane>

<script>
import common from "../common/Config";
//引入xterm终端依赖
import { Terminal } from 'xterm';
import { FitAddon } from 'xterm-addon-fit';
import 'xterm/css/xterm.css';
import 'xterm/lib/xterm.js';
export default {
    data() {
        return {
            //terminal
            term: null,
            socket: null
        }
    },
    methods: {
        initTerm() {
            //初始化xterm实例
            this.term = new Terminal({
                rendererType: 'canvas', //渲染类型
                rows: 30, //行数
                cols: 110,
                convertEol: false, //启用时，光标将设置为下一行的开头
                scrollback: 10, //终端中的回滚量
                disableStdin: false, //是否应禁用输入
                cursorStyle: 'underline', //光标样式
                cursorBlink: true, //光标闪烁
                theme: {
                foreground: 'white', //字体
                background: '#060101', //背景色
                cursor: 'help' //设置光标
                }
            });
            //绑定dom
            this.term.open(document.getElementById('xterm'))
            //终端适应父元素大小
            const fitAddon = new FitAddon()
            this.term.loadAddon(fitAddon)
            fitAddon.fit();
            //获取终端的焦点
            this.term.focus();
            let _this = this; //一定要重新定义一个this，不然this指向会出问题
            //onData方法用于定义输入的动作
            this.term.onData(function (key) {
                // 这里key值是输入的值，数据格式就是后端定义的
{"operation":"stdin","data":"ls"}
                let msgOrder = {
                operation: 'stdin',
                data: key,
                };
                //发送数据
                _this.socket.send(JSON.stringify(msgOrder));
            });
            //发送resize请求
            let msgOrder2 = {
                operation: 'resize',
                cols: this.term.cols,
                rows: this.term.rows,
            };
            this.socket.send(JSON.stringify(msgOrder2))
        },
        //初始化websocket
        initSocket(row) {
            //定义websocket连接地址
            let terminalWsUrl = common.k8sTerminalWs + "?pod_name=" +
row.metadata.name + "&container_name=" + this.containerValue + "&namespace=" +
this.namespaceValue
            //实例化
            this.socket = new WebSocket(terminalWsUrl);
            //关闭连接时的方法
            this.socketOnClose();
            //建立连接时的方法
            this.socketOnOpen();
            //接收消息的方法
            this.socketOnMessage();
            //报错时的方法
            this.socketOnError();
        },
        socketOnOpen() {
            this.socket.onopen = () => {
                //简历连接成功后，初始化虚拟终端
                this.initTerm()
            }
        },
        socketOnMessage() {
            this.socket.onmessage = (msg) => {
                //接收到消息后将字符串转为对象，输出data内容
                let content = JSON.parse(msg.data)
                this.term.write(content.data)
            }
        },
        socketOnClose() {
            this.socket.onclose = () => {
                //关闭连接后打印在终端里
                this.term.write("链接已关闭")
            }
        },
        socketOnError() {
            this.socket.onerror = () => {
                console.log('socket 链接失败')
            }
        },
        //关闭连接
        closeSocket() {
            //若没有实例化，则不需要关闭
            if (this.socket === null) {
                    return
                }
            this.term.write("链接关闭中。。。")
            this.socket.close()
        }
    },
    beforeUnmount() {
        //若websocket连接没有关闭，则在改生命周期关闭
        if ( this.socket !== null ) {
            this.socket.close()
        }
    },
}
</script>
```

2.3 DaemonSet

（1）功能

列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

daemonset信息

```
<el-table-column width="20"></el-table-column>
<!-- DaemonSet名字 -->
<el-table-column align=left label="DaemonSet名">
    <!-- 插槽，scope.row获取当前行的数据 -->
    <template v-slot="scope">
        <a class="daemonset-body-daemonsetname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<!-- 标签 -->
<el-table-column align=center label="标签">
    <template v-slot="scope">
        <!-- for循环，每个label只显示固定长度，鼠标悬停后气泡弹出框显示完整长度 -->
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <!-- 气泡弹出框 -->
            <!-- placement 弹出位置 -->
            <!-- trigger 触发条件 -->
            <!-- content 弹出框内容 -->
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
                    <!-- ellipsis方法用于剪裁字符串 -->
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
            </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<!-- 容器组 -->
<el-table-column align=center label="容器组">
    <!-- 可用数量/总数量，三元运算，若值大于0则显示值，否则显示0 -->
    <template v-slot="scope">
        <span>{{ scope.row.status.numberAvailable>0?scope.row.status.numberAvailable:0
}} / {{ scope.row.status.desiredNumberScheduled>0?
scope.row.status.desiredNumberScheduled:0 }} </span>
    </template>
</el-table-column>
<!-- 创建时间 -->
<el-table-column align=center min-width="100" label="创建时间">
```

```
    <!-- timeTrans函数用于将格林威治时间转成北京时间 -->
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
<!-- 容器镜像 -->
<el-table-column align=center label="镜像">
    <!-- 与label的显示逻辑一致 -->
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.spec.template.spec.containers" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="val.image">
                <template #reference>
                    <el-tag style="margin-bottom: 5px">{{
ellipsis(val.image.split('/')[2]==undefined?val.image:val.image.split('/')[2]) }}</el-
tag>
            </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
```

2.4 StatefulSet

(1) 功能

列表、详情、更新、删除

(2) 布局

(3) 头部工具栏

(4) 数据表格

statefulset信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="StatefulSet名">
    <template v-slot="scope">
        <a class="statefulset-body-statefulsetname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签">
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
```

```
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
            </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<el-table-column align=center label="容器组">
    <template v-slot="scope">
        <span>{{ scope.row.status.currentReplicas>0?scope.row.status.currentReplicas:0
}} / {{ scope.row.spec.replicas>0?scope.row.spec.replicas:0 }} </span>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
<el-table-column align=center label="镜像">
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.spec.template.spec.containers" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="val.image">
                <template #reference>
                    <el-tag style="margin-bottom: 5px">{{
ellipsis(val.image.split('/')[2]==undefined?val.image:val.image.split('/')[2]) }}</el-
tag>
                </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
```

3、集群

3.1 Node

（1）功能

列表、详情、更新

（2）布局

（3）头部工具栏

（4）数据表格

node信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="Node名">
```

```
    <template v-slot="scope">
    <p class="node-body-nodename">{{ scope.row.metadata.name }}</p>
    <p class="node-body-ip">{{ scope.row.status.addresses[0].address }}</p>
    </template>
</el-table-column>
<el-table-column align=center label="规格">
    <template v-slot="scope">
        <el-tag type="warning">{{ scope.row.status.capacity.cpu }}核{{
specTrans(scope.row.status.capacity.memory) }}G</el-tag>
    </template>
</el-table-column>
<el-table-column align=center label="POD-CIDR">
    <template v-slot="scope">
        <span>{{ scope.row.spec.podCIDR }} </span>
    </template>
</el-table-column>
<el-table-column align=center label="版本">
    <template v-slot="scope">
        <span>{{ scope.row.status.nodeInfo.kubeletVersion }} </span>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
```

3.2 Namespace

（1）功能

列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

namespace信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="Namespace名">
    <template v-slot="scope">
<a class="namespace-body-namespacename">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签" min-width='120'>
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
```

```
                  <template #reference>
                      <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
                  </template>
              </el-popover>
          </div>
      </template>
  </el-table-column>
  <el-table-column align=center prop="status.phase" label="状态">
      <template v-slot="scope">
          <span :class="[scope.row.status.phase === 'Active' ? 'success-status' :
'error-status']">{{ scope.row.status.phase }}       </span>
      </template>
  </el-table-column>
  <el-table-column align=center min-width="100" label="创建时间">
      <template v-slot="scope">
          <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
      </template>
  </el-table-column>
```

3.3 PV

（1）功能

　列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

pv信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="PV名">
    <template v-slot="scope">
        <a class="pv-body-pvname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="状态">
    <template v-slot="scope">
        <span :class="[scope.row.status.phase === 'Bound' ? 'success-status' : 'error-
status']">{{ scope.row.status.phase }}</span>
    </template>
</el-table-column>
<el-table-column align=center prop="spec.accessModes[0]" label="访问模式"></el-table-
column>
<el-table-column align=center prop="spec.capacity.storage" label="容量"></el-table-
column>
<el-table-column align=center prop="spec.claimRef.name" label="PVC"></el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
```

```
    </template>
</el-table-column>
```

4、负载均衡

4.1 Service

（1）功能

列表、详情、新增、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

service信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="Service名">
    <template v-slot="scope">
        <a class="service-body-servicename">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签" min-width='120'>
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
                </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<el-table-column align=center label="类型">
    <template v-slot="scope">
        <span style="font-weight:bold;">{{ scope.row.spec.type }} </span>
    </template>
</el-table-column>
<el-table-column align=center label="CLUSTER-IP">
    <template v-slot="scope">
        <span>{{ scope.row.spec.clusterIP }} </span>
    </template>
</el-table-column>
<el-table-column align=center label="EXTERNAL-IP">
    <template v-slot="scope">
<span>{{ scope.row.status.loadBalancer.ingress ?
scope.row.status.loadBalancer.ingress[0].ip : '' }} </span>
```

```
    </template>
</el-table-column>
<el-table-column align=center label="端口">
    <template v-slot="scope">
        <span v-if="!scope.row.spec.ports[0].nodePort">{{ scope.row.spec.ports[0].port
}}/{{ scope.row.spec.ports[0].protocol }}</span>
        <span v-if="scope.row.spec.ports[0].nodePort">{{ scope.row.spec.ports[0].port
}}:{{ scope.row.spec.ports[0].nodePort }}/{{ scope.row.spec.ports[0].protocol }}
</span>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
```

创建Drawer

```
<el-drawer
           v-model="createServiceDrawer"
           :direction="direction"
           :before-close="handleClose">
    <template #title>
        <h4>创建Service</h4>
    </template>
    <template #default>
        <el-row type="flex" justify="center">
            <el-col :span="20">
            <el-form ref="createService" :rules="createServiceRules"
:model="createService" label-width="80px">
                <el-form-item class="service-create-form" label="名称" prop="name">
                    <el-input v-model="createService.name"></el-input>
                </el-form-item>
                <el-form-item class="service-create-form" label="命名空间"
prop="namespace">
                    <el-select v-model="createService.namespace" filterable
placeholder="请选择">
                        <el-option
                                   v-for="(item, index) in namespaceList"
                                   :key="index"
                                   :label="item.metadata.name"
                                   :value="item.metadata.name">
                        </el-option>
                    </el-select>
                </el-form-item>v
                <el-form-item class="service-create-form" label="类型" prop="type">
                    <el-select v-model="createService.type" placeholder="请选择">
                        <el-option value="ClusterIP" label="ClusterIP"></el-option>
                        <el-option value="NodePort" label="NodePort"></el-option>
                    </el-select>
                </el-form-item>
                <el-form-item class="deploy-create-form" label="容器端口"
prop="container_port">
```

```
                    <el-input v-model="createService.container_port" placeholder="示例：
80"></el-input>
                </el-form-item>
                <el-form-item class="service-create-form" label="Service端口"
prop="port">
                    <el-input v-model="createService.port" placeholder="示例：80"></el-
input>
                </el-form-item>
                <el-form-item v-if="createService.type == 'NodePort'" class="service-
create-form" label="NodePort" prop="node_port">
                    <el-input v-model="createService.node_port" placeholder="示例：
30001"></el-input>
                </el-form-item>
                <el-form-item class="SERVICE-create-form" label="标签"
prop="label_str">
                    <el-input v-model="createService.label_str" placeholder="示例：
project=ms,app=gateway"></el-input>
                </el-form-item>
            </el-form>
            </el-col>
        </el-row>
    </template>
    <template #footer>
        <el-button @click="createServiceDrawer = false">取消</el-button>
        <el-button type="primary" @click="submitForm('createService')">立即创建</el-
button>
    </template>
</el-drawer>
```

4.2 Ingress

（1）功能

　列表、详情、新增、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

ingress信息

```
<el-table-column width="10"></el-table-column>
<el-table-column align=left label="Ingress名">
    <template v-slot="scope">
        <a class="ingress-body-ingressname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签" min-width='120'>
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
                </template>
            </el-popover>
        </div>
</template>
</el-table-column>
<el-table-column align=center label="Host" min-width='120'>
    <template v-slot="scope">
        <div v-for="(item, index) in scope.row.spec.rules" :key="index">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="item.host">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="danger">{{
ellipsis(item.host) }}</el-tag>
                </template>
            </el-popover>
        </div>
</template>
</el-table-column>
<el-table-column align=center label="Path">
    <template v-slot="scope">
        <div v-for="(item, index) in scope.row.spec.rules" :key="index">
            <el-popover
                        placement="right"
                        :width="100"
                        trigger="hover"
                        :content="item.http.paths[0].path">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="danger">{{
item.http.paths[0].path }}</el-tag>
                </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<el-table-column align=center label="EXTERNAL-IP">
    <template v-slot="scope">
        <span>{{ scope.row.status.loadBalancer.ingress ?
scope.row.status.loadBalancer.ingress[0].ip : '' }} </span>
    </template>
</el-table-column>
<el-table-column align=center label="TLS">
    <template v-slot="scope">
        <span>{{ scope.row.spec.tls ? 'YES' : '' }} </span>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
```

创建Drawer

```
<el-drawer
           v-model="createIngressDrawer"
           :direction="direction"
           :before-close="handleClose">
    <template #title>
        <h4>创建Ingress</h4>
    </template>
    <template #default>
        <el-row type="flex" justify="center">
            <el-col :span="20">
                <el-form ref="createIngress" :rules="createIngressRules"
:model="createIngress" label-width="80px">
                    <el-form-item class="ingress-create-form" label="名称" prop="name">
                        <el-input v-model="createIngress.name"></el-input>
                    </el-form-item>
                    <el-form-item class="ingress-create-form" label="命名空间"
prop="namespace">
                        <el-select v-model="createIngress.namespace" filterable
placeholder="请选择">
                            <el-option
                                       v-for="(item, index) in namespaceList"
                                       :key="index"
                                       :label="item.metadata.name"
                                       :value="item.metadata.name">
                            </el-option>
                        </el-select>
                    </el-form-item>
                    <el-form-item class="SERVICE-create-form" label="标签"
prop="label_str">
                        <el-input v-model="createIngress.label_str" placeholder="示例:
project=ms,app=gateway"></el-input>
                    </el-form-item>
                    <el-form-item class="deploy-create-form" label="域名" prop="host">
                        <el-input v-model="createIngress.host" placeholder="示例:
www.example.com"></el-input>
                    </el-form-item>
                    <el-form-item class="ingress-create-form" label="Path"
prop="path">
                        <el-input v-model="createIngress.path" placeholder="示例:
/abc"></el-input>
                    </el-form-item>
                    <el-form-item class="deploy-create-form" label="匹配类型"
prop="path_type">
                        <el-select v-model="createIngress.path_type" placeholder="请选
择">
                            <el-option value="Prefix" label="Prefix"></el-option>
                            <el-option value="Exact" label="Exact"></el-option>
                            <el-option value="ImplementationSpecific"
label="ImplementationSpecific"></el-option>
                        </el-select>
                    </el-form-item>
                    <el-form-item class="ingress-create-form" label="Service名"
prop="service_name">
```

```
                        <el-input disabled v-model="createIngress.name"></el-input>
                    </el-form-item>
                    <el-form-item class="ingress-create-form" label="Service端口" prop="service_port">
                        <el-input v-model="createIngress.service_port" placeholder="示例: 80"></el-input>
                    </el-form-item>
                </el-form>
            </el-col>
        </el-row>
    </template>
    <template #footer>
        <el-button @click="createIngressDrawer = false">取消</el-button>
        <el-button type="primary" @click="submitForm('createIngress')">立即创建</el-button>
    </template>
</el-drawer>
```

> 阿良教育： **www.aliangedu.cn**

5、存储与配置

5.1 ConfigMap

（1）功能

列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

config信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="ConfigMap名">
    <template v-slot="scope">
        <a class="configmap-body-configmapname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签">
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key + ":" + val) }}</el-tag>
                </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<el-table-column align=center label="DATA">
    <template v-slot="scope">
        <el-popover
                style="overflow:auto"
                placement="right"
                :width="400"
                trigger="click">
            <div style="overflow-y:auto;max-height:500px;">
                <span>{{ scope.row.data }}</span>
            </div>
            <template #reference>
                <el-icon style="font-size:18px;cursor:pointer;"><reading/></el-icon>
            </template>
        </el-popover>
    </template>
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
<el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }} </el-tag>
    </template>
</el-table-column>
```

5.2 Secret

（1）功能

列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

secret信息

```
<el-table-column width="20"></el-table-column>
<el-table-column align=left label="Secret名">
    <template v-slot="scope">
        <a class="secret-body-secretname">{{ scope.row.metadata.name }}</a>
    </template>
</el-table-column>
<el-table-column align=center label="标签">
    <template v-slot="scope">
        <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
            <el-popover
                        placement="right"
                        :width="200"
                        trigger="hover"
                        :content="key + ':' + val">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
                </template>
```

```
              </el-popover>
          </div>
          </template>
  </el-table-column>
  <el-table-column align=center label="DATA">
      <template v-slot="scope">
          <el-popover
                  style="overflow:auto"
                  placement="right"
                  :width="400"
                  trigger="click">
              <div style="overflow-y:auto;max-height:500px;">
                  <span>{{ scope.row.data }}</span>
              </div>
              <template #reference>
                  <el-icon style="font-size:18px;cursor:pointer;"><reading/></el-icon>
              </template>
          </el-popover>
      </template>
  </el-table-column>
  <el-table-column align=center prop="type" min-width="100" label="类型">
  </el-table-column>
  <el-table-column align=center min-width="100" label="创建时间">
      <template v-slot="scope">
          <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
  </el-tag>
      </template>
  </el-table-column>
```

5.3 PVC

（1）功能

列表、详情、更新、删除

（2）布局

（3）头部工具栏

（4）数据表格

pvc信息

```
  <el-table-column width="20"></el-table-column>
  <el-table-column align=left label="PVC名">
      <template v-slot="scope">
          <a class="pvc-body-pvcname">{{ scope.row.metadata.name }}</a>
      </template>
  </el-table-column>
  <el-table-column align=center label="标签">
      <template v-slot="scope">
          <div v-for="(val, key) in scope.row.metadata.labels" :key="key">
              <el-popover
                          placement="right"
                          :width="200"
                          trigger="hover"
```

```
                        :content="key + ':' + val">
                <template #reference>
                    <el-tag style="margin-bottom: 5px" type="warning">{{ ellipsis(key
+ ":" + val) }}</el-tag>
                </template>
            </el-popover>
        </div>
    </template>
</el-table-column>
<el-table-column align=center label="状态">
    <template v-slot="scope">
        <span :class="[scope.row.status.phase === 'Bound' ? 'success-status' : 'error-
status']">{{ scope.row.status.phase }}</span>
    </template>
</el-table-column>
<el-table-column align=center prop="status.capacity.storage" label="容量">
</el-table-column>
<el-table-column align=center prop="status.accessModes[0]" label="访问模式">
</el-table-column>
<el-table-column align=center prop="spec.storageClassName" label="StorageClass">
</el-table-column>
<el-table-column align=center min-width="100" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTrans(scope.row.metadata.creationTimestamp) }}
</el-tag>
    </template>
</el-table-column>
```

6、概要

（1）布局

```
<template>
    <div class="home">
        <!-- 折叠面板 -->
        <el-collapse v-model="activeNames">
            <!-- 面板1 集群资源卡片 -->
            <el-collapse-item title="集群资源" name="1">
            </el-collapse-item>
            <!-- 面板2 节点资源卡片 -->
            <el-collapse-item title="节点资源" name="2">
            </el-collapse-item>
            <!-- 面板3 资源统计画图 -->
            <el-collapse-item title="资源统计" name="3">
            </el-collapse-item>
        </el-collapse>
    </div>
</template>

<script>
export default {
    data() {
        return {
            //控制折叠面板的展开，表示打开所有的折叠面板
            activeNames: ["1", "2", "3"],
        }
    }
}
</script>

<style scoped>
    /deep/ .el-collapse-item__header {
        font-size: 16px;
    }
</style>
```

(2) 状态展示框

```
<!-- 面板2 节点资源卡片 -->
<el-collapse-item title="节点资源" name="2">
    <el-row :gutter="10" style="margin-bottom: 10px;">
        <!-- 节点数量 -->
        <el-col :span="5">
            <el-card class="home-node-card" :body-style="{padding:'10px'}">
                <div style="float:left;padding-top:20%">
                    <el-progress :stroke-width="20" :show-text="false" type="circle"
:percentage="nodeTotal/nodeTotal * 100"></el-progress>
                </div>
                <div>
                    <p class="home-node-card-title">节点: Ready/总数量</p>
                    <p class="home-node-card-num">{{ nodeTotal }}/{{ nodeTotal }}</p>
                </div>
            </el-card>
        </el-col>
    </el-row>
</el-collapse-item>

<script>
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //获取node的属性
            nodeTotal: 0,
        }
    },
    methods: {
        //获取node属性
        getNodes() {
            httpClient.get(this.getNodesData.url, {params: this.getNodesData.params})
            .then(res => {
                this.nodeTotal = res.data.total
                let nodeList = res.data.items
                let index
                for (index in nodeList) {
                    //正则匹配纯数字，如果不是纯数字则跳过
                    let isnum = /^\d+$/.test(nodeList[index].status.allocatable.cpu);
                    if (!isnum) {
                        continue
                    }
                    //计算node的cpu mem和pod的可分配及总容量数据
                    this.nodeCpuAllocatable =
parseInt(nodeList[index].status.allocatable.cpu) + this.nodeCpuAllocatable
                    this.nodeCpuCapacity =
parseInt(nodeList[index].status.capacity.cpu) + this.nodeCpuCapacity
                    this.nodeMemAllocatable =
parseInt(nodeList[index].status.allocatable.memory) + this.nodeMemAllocatable
                    this.nodeMemCapacity =
parseInt(nodeList[index].status.capacity.memory) + this.nodeMemCapacity
                    this.nodePodAllocatable =
parseInt(nodeList[index].status.allocatable.pods) + this.nodePodAllocatable
                    this.nodePodCapacity =
parseInt(nodeList[index].status.capacity.pods) + this.nodePodCapacity
                }
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        }
    },
    beforeMount() {
        this.getNodes()
    }
}
</script>

<style scoped>
    /deep/ .el-collapse-item__header {
        font-size: 16px;
    }
    .home-node-card {
        border-radius:1px;
        text-align: center;
        background-color: rgb(250, 253, 255);
    }
    .home-node-card-title {
        font-size: 12px;
    }
    .home-node-card-num {
        font-size: 22px;
        font-weight: bold;
        color: rgb(63, 92, 135);
    }
    /deep/ .el-progress-circle {
        height: 50px !important;
        width: 50px !important;
    }
</style>
```

(3) 数据统计图

```
<!-- 面板3 资源统计画图 -->
<el-collapse-item title="资源统计" name="3">
    <el-row :gutter="10">
```

```
        <!-- 每个namspace中pod数量的作图统计 -->
        <el-col :span="24" style="margin-bottom: 10px;">
            <el-card class="home-dash-card" :body-style="{padding:'10px'}">
                <!-- 这个div就是画图的内容，echarts初始化后会绑定到这个id上展示出来 -->
                <div id="podNumDash" style="height: 300px;">
                </div>
            </el-card>
        </el-col>
    </el-row>
</el-collapse-item>

<script>
//引入echarts
import * as echarts from 'echarts'
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            //每个namespace中pod的数量[{namespace:"default",pod_num:5}]
            podNumNp: [],
            podNumNpUrl: common.k8sPodNumNp,
            podNumDash: null
        }
    },
    methods: {
        //获取每个namespace中pod的数量
        getPodNumNp() {
            httpClient.get(this.podNumNpUrl)
            .then(res => {
                this.podNumNp = res.data
                //echarts作图
                this.getPodNumDash()
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        },
        getPodNumDash(){
            //若实例已经初始化了，则销毁实例
            if (this.podNumDash != null && this.podNumDash != "" && this.podNumDash !=
undefined) {
                this.podNumDash.dispose()
            }
            //初始化实例，绑定到dom上
            this.podNumDash = echarts.init(document.getElementById('podNumDash'));
            //echarts作图配置
            this.podNumDash.setOption({
                //标题及字体颜色
                title: { text: 'Pods per Namespace', textStyle: {color:'rgb(134, 135,
136)'}},
                //图表颜色
                color: ['#67E0E3', '#9FE6B8', '#FFDB5C','#ff9f7f', '#fb7293',
'#E062AE', '#E690D1', '#e7bcf3', '#9d96f5', '#8378EA', '#96BFFF'],
                //提示框
                tooltip: {
```

```
                //触发类型坐标轴触发
                trigger: "axis",
                //'cross' 十字准星指示器
                axisPointer: {
                    type: "cross",
                    label: {
                        backgroundColor: "#76baf1"
                    }
                }
            },
            //图表中的数据类型解释
            legend: {
                data: ['Pods']
            },
            //图表数据集
            dataset: {
                //维度定义，默认第一个元素表示x轴的数据，其他都是y轴数据
                dimensions: ['namespace','pod_num'],
                //源数据
                source: this.podNumNp
            },
            //x轴属性
            xAxis: {
                //category类目轴，value数值轴，time时间轴，log对数轴
                type: 'category',
                //轴标签
                axisLabel:{
                    //坐标轴刻度标签的显示间隔，在类目轴中有效.0显示所有
                    interval: 0,
                    //格式化轴标签
                    formatter: function (value) {
                        return value.length>5?value.substring(0,5)+'...':value
                    }
                },
            },
            //y轴属性
            yAxis: [
                //数值轴
                {type: 'value'}
            ],
            //定义系列，用于指定一组数值以及他们映射成的图
            series: [{
                //name是legend对应的值
                name: 'Pods',
                //bar柱状图，line折线图，pie饼图等等
                type: 'bar',
                //每个类目的值标签，配置
                label: {
                    //是否显示值
                    show: true,
                    //显示的位置
                    position: 'top'
                    }
                }
            ]
        });
    }
},
    beforeMount() {
        this.getPodNumNp()
    }
}
</script>

<style scoped>
    .home-dash-card {
        border-radius:1px;
    }
</style>
```

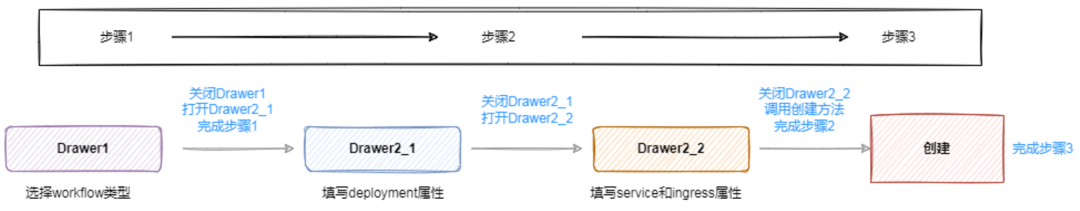
7、工作流

(1) 功能

(2) 布局

(3) 头部工具栏

(4) 步骤条



抽屉弹出框1

```
<template>
    <div class="workflow">
        <el-row>
            <!-- header2 步骤条 -->
            <el-col :span="24">
                <div>
                    <!-- 步骤条展示，active属性控制到了哪一步 -->
                    <el-card class="workflow-head-card" shadow="never" :body-style="
{padding:'30px 10px 20px 10px'}">
                        <el-steps :active="active" align-center finish-
status="success">
                            <el-step title="步骤1" description="选择工作流类型, ClusterIP
NodePort Workflow"></el-step>
                            <el-step title="步骤2" description="填写Deployment Workflow
Workflow表单"></el-step>
                            <el-step title="步骤3" description="创建Deployment Workflow
Workflow"></el-step>
                        </el-steps>
                    </el-card>
                </div>
            </el-col>
            <!-- header3 -->
            <el-col :span="24">
```

```
                <div>
                    <el-card class="workflow-head-card" shadow="never" :body-style="
{padding:'10px'}">
                        <el-row>
                            <el-col :span="3">
                                <div>
                                    <!-- 创建工作流 -->
                                    <!-- createWorkflowDrawerIndex1-》
createWorkflowDrawerIndex2-1-》createWorkflowDrawerIndex2-2 -->
                                    <el-button style="border-radius:2px;" icon="Edit"
type="primary" @click="createWorkflowDrawerIndex1 = true" v-
loading.fullscreen.lock="fullscreenLoading">创建工作流</el-button>
                                </div>
                            </el-col>
                            <el-col :span="6">
                                <div>
                                    <el-input class="workflow-head-search" clearable
placeholder="请输入" v-model="searchInput"></el-input>
                                    <el-button style="border-radius:2px;"
icon="Search" type="primary" plain @click="getWorkflows()">搜索</el-button>
                                </div>
                            </el-col>
                        </el-row>
                    </el-card>
                </div>
            </el-col>
        </el-row>
    <!-- 抽屉弹框1 -->
    <el-drawer
        v-model="createWorkflowDrawerIndex1"
        :direction="direction"
        :before-close="handleClose">
        <template #title>
            <h4>创建Workflow-步骤1</h4>
        </template>
        <template #default>
            <el-row type="flex" justify="center">
                <el-col :span="20">
                    <el-form label-width="80px">
                        <el-form-item class="workflow-create-form" label="类型"
prop="name">
                            <el-radio v-model="createWorkflow.type"
label="ClusterIP">ClusterIP</el-radio>
                            <el-radio v-model="createWorkflow.type"
label="NodePort">NodePort</el-radio>
                            <el-radio v-model="createWorkflow.type"
label="Ingress">Ingress</el-radio>
                        </el-form-item>
                    </el-form>
                </el-col>
            </el-row>
        </template>
        <template #footer>
            <el-button @click="drawerCancel('createWorkflowDrawerIndex1')">取消</el-
button>
            <el-button type="primary" @click="workflowIndex1Next()">下一步</el-button>
        </template>
    </el-drawer>
    </div>
</template>

<script>
export default {
    data() {
        return {
            //工作流以及3个抽屉弹出框
            active: 0,
            createWorkflowDrawerIndex1: false,
            createWorkflowDrawerIndex2_1: false,
            createWorkflowDrawerIndex2_2: false,
            fullscreenLoading: false,
            direction: 'rtl',
            createWorkflow: {
                name: '',
                namespace: '',
                replicas: 1,
                image: '',
                resource: '',
                health_check: false,
                health_path: '',
                label_str: '',
                label: {},
                container_port: '',
                type: '',
                port: '',
                node_port: '',
                host: '',
                path: '',
                path_type: ''
            },
        }
    },
    methods: {
        handleClose(done) {
            this.$confirm('确认关闭？')
            .then(() => {
                done();
            })
            .catch(() => {});
            this.active = 0
        },
        //关闭抽屉
        drawerCancel(drawerName) {
            switch (drawerName) {
                case 'createWorkflowDrawerIndex1':
                    this.createWorkflowDrawerIndex1 = false
                    break
                case 'createWorkflowDrawerIndex2_1':
                    this.createWorkflowDrawerIndex2_1 = false
                    break
                case 'createWorkflowDrawerIndex2_2':
                    this.createWorkflowDrawerIndex2_2 = false
            }
            this.active = 0
        },
        //抽屉1的提交
        workflowIndex1Next() {
            //判断是否选择了type
            if (!this.createWorkflow.type) {
                this.$message.warning({
                    message: "请选择工作流类型"
                })
                return
            }
            //关闭抽屉1
            this.createWorkflowDrawerIndex1 = false
            //打开抽屉2_1
            this.createWorkflowDrawerIndex2_1 = true
            //步骤条完成第一步
            this.active = 1
        }
    }
}
</script>

<style scoped>
    /deep/ .el-drawer__header {
        margin-bottom: 0px !important;
    }
    /deep/ .el-drawer__body {
        padding: 0px 0px 0px 0px;
    }
</style>
```

抽屉弹出框2

```
    <!-- 抽屉弹框2 -->
    <el-drawer
        v-model="createWorkflowDrawerIndex2_1"
        :direction="direction"
        :before-close="handleClose">
        <template #title>
            <h4>创建Workflow-步骤2</h4>
        </template>
        <template #default>
            <el-row type="flex" justify="center">
                <el-col :span="20">
                    <el-form ref="createWorkflow" :rules="createWorkflowRules"
:model="createWorkflow" label-width="80px">
                        <h4 style="margin-bottom:10px">Deployment</h4>
                        <el-form-item class="workflow-create-form" label="名称"
prop="name">
                            <el-input v-model="createWorkflow.name"></el-input>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="命名空间"
prop="namespace">
                            <el-select v-model="createWorkflow.namespace" filterable
placeholder="请选择">
                                <el-option
                                v-for="(item, index) in namespaceList"
                                :key="index"
                                :label="item.metadata.name"
                                :value="item.metadata.name">
                                </el-option>
                            </el-select>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="副本数"
prop="replicas">
                            <el-input-number v-model="createWorkflow.replicas"
:min="1" :max="10"></el-input-number>
                                <el-popover
                                    placement="top"
                                    :width="100"
                                    trigger="hover"
                                    content="申请副本数上限为10个">
                                    <template #reference>
                                        <el-icon style="width:2em;font-
size:18px;color:#4795EE"><WarningFilled/></el-icon>
                                    </template>
                                </el-popover>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="镜像"
prop="image">
                            <el-input v-model="createWorkflow.image"></el-input>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="标签"
prop="label_str">
                            <el-input v-model="createWorkflow.label_str"
placeholder="示例: project=ms,app=gateway"></el-input>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="资源配额"
prop="resource">
                            <el-select v-model="createWorkflow.resource"
placeholder="请选择">
                                <el-option value="0.5/1" label="0.5C1G"></el-option>
                                <el-option value="1/2" label="1C2G"></el-option>
                                <el-option value="2/4" label="2C4G"></el-option>
                                <el-option value="4/8" label="4C8G"></el-option>
                            </el-select>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="容器端口"
prop="container_port">
                            <el-input v-model="createWorkflow.container_port"
placeholder="示例: 80"></el-input>
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="健康检查"
prop="health">
                            <el-switch v-model="createWorkflow.health_check" />
                        </el-form-item>
                        <el-form-item class="workflow-create-form" label="检查路径"
prop="healthPath">
                            <el-input v-model="createWorkflow.health_path"
placeholder="示例: /health"></el-input>
                        </el-form-item>
                    </el-form>
                </el-col>
            </el-row>
        </template>
        <template #footer>
```

```
            <el-button @click="drawerCancel('createWorkflowDrawerIndex2_1')">取消</el-
button>
            <el-button type="primary" @click="submitForm('createWorkflow',
workflowIndex2_1Next)">下一步</el-button>
        </template>
    </el-drawer>

<script>
export default {
    data() {
        return {
            createWorkflowRules: {
                name: [{
                    required: true,
                    message: '请填写名称',
                    trigger: 'change'
                }],
                image: [{
                    required: true,
                    message: '请填写镜像',
                    trigger: 'change'
                }],
                namespace: [{
                    required: true,
                    message: '请选择命名空间',
                    trigger: 'change'
                }],
                resource: [{
                    required: true,
                    message: '请选择配额',
                    trigger: 'change'
                }],
                label_str: [{
                    required: true,
                    message: '请填写标签',
                    trigger: 'change'
                }],
                container_port: [{
                    required: true,
                    message: '请填写容器端口',
                    trigger: 'change'
                }],
                type: [{
                    required: true,
                    message: '请填写工作流类型',
                    trigger: 'change'
                }],
                port: [{
                    required: true,
                    message: '请填写Workflow端口',
                    trigger: 'change'
                }],
                node_port: [{
                    required: true,
                    message: '请填写NodePort',
                    trigger: 'change'
                }],
                host: [{
                    required: true,
                    message: '请填写域名',
                    trigger: 'change'
                }],
                path: [{
                    required: true,
                    message: '请填写路径',
                    trigger: 'change'
                }],
                path_type: [{
                    required: true,
                    message: '你选择匹配类型',
                    trigger: 'change'
                }],
            }
        }
    },
    methods: {
        //抽屉2_2提交
        submitForm(formName, fn) {
            this.$refs[formName].validate((valid) => {
                if (valid) {
                    fn()
                } else {
                    return false;
                }
            })
        },
        //抽屉2的提交
        workflowIndex2_1Next() {
            //关闭抽屉2_1
            this.createWorkflowDrawerIndex2_1 = false
            //打开抽屉2_2
            this.createWorkflowDrawerIndex2_2 = true
        }
    }
}
</script>
```

抽屉弹出框3

```
      <!-- 抽屉弹框3 -->
      <el-drawer
          v-model="createWorkflowDrawerIndex2_2"
          :direction="direction"
          :before-close="handleClose">
          <template #title>
              <h4>创建Workflow-步骤2</h4>
          </template>
          <template #default>
              <el-row type="flex" justify="center">
                  <el-col :span="20">
                      <el-form ref="createWorkflow" :rules="createWorkflowRules"
  :model="createWorkflow" label-width="80px">
                          <h4 style="margin-bottom:10px">Service</h4>
```

```
                        <el-form-item class="service-create-form" label="Service端口"
prop="port">
                            <el-input v-model="createWorkflow.port" placeholder="示例:
80"></el-input>
                        </el-form-item>
                        <el-form-item v-if="createWorkflow.type == 'NodePort'"
class="service-create-form" label="NodePort" prop="node_port">
                            <el-input v-model="createWorkflow.node_port"
placeholder="示例: 30001"></el-input>
                        </el-form-item>
                        <el-divider v-if="createWorkflow.type == 'Ingress'"></el-
divider>
                        <h4 v-if="createWorkflow.type == 'Ingress'" style="margin-
bottom:10px">Ingress</h4>
                        <el-form-item v-if="createWorkflow.type == 'Ingress'"
class="deploy-create-form" label="域名" prop="host">
                            <el-input v-model="createWorkflow.host" placeholder="示例:
www.example.com"></el-input>
                        </el-form-item>
                        <el-form-item v-if="createWorkflow.type == 'Ingress'"
class="ingress-create-form" label="Path" prop="path">
                            <el-input v-model="createWorkflow.path" placeholder="示例:
/abc"></el-input>
                        </el-form-item>
                        <el-form-item v-if="createWorkflow.type == 'Ingress'"
class="deploy-create-form" label="匹配类型" prop="path_type">
                            <el-select v-model="createWorkflow.path_type"
placeholder="请选择">
                                <el-option value="Prefix" label="Prefix"></el-option>
                                <el-option value="Exact" label="Exact"></el-option>
                                <el-option value="ImplementationSpecific"
label="ImplementationSpecific"></el-option>
                            </el-select>
                        </el-form-item>
                    </el-form>
                </el-col>
            </el-row>
        </template>
        <template #footer>
            <el-button @click="drawerCancel('createWorkflowDrawerIndex2_2')">取消</el-
button>
            <el-button type="primary" @click="submitForm('createWorkflow',
createWorkflowFunc)">立即创建</el-button>
        </template>
    </el-drawer>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
export default {
    data() {
        return {
            createWorkflowData: {
                url: common.k8sWorkflowCreate,
                params: {}
            }
        }
    },
    methods: {
        //真正的创建workflow的方法
        createWorkflowFunc() {
            //验证标签,如果不符合a=b,c=d的格式，咱返回
            let reg = new RegExp("(^[A-Za-z]+=[A-Za-z0-9]+).*")
            if (!reg.test(this.createWorkflow.label_str)) {
                this.$message.warning({
                    message: "标签填写异常，请确认后重新填写"
                })
                return
            }
            //加载动画开启
            this.fullscreenLoading = true
            //处理标签,将标签转成map   a=b  ->  map[a]=b
            let label = new Map()
            let cpu, memory
            let a = (this.createWorkflow.label_str).split(",")
            a.forEach(item => {
                let b = item.split("=")
                label[b[0]] = b[1]
            })
            //处理配额
            let resourceList = this.createWorkflow.resource.split("/")
            cpu = resourceList[0]
            memory = resourceList[1] + "Gi"
            //处理其他参数
            this.createWorkflowData.params = this.createWorkflow
            this.createWorkflowData.params.label = label
            this.createWorkflowData.params.cpu = cpu
            this.createWorkflowData.params.memory = memory
            this.createWorkflowData.params.container_port =
parseInt(this.createWorkflow.container_port)
            this.createWorkflowData.params.port = parseInt(this.createWorkflow.port)
            this.createWorkflowData.params.node_port =
parseInt(this.createWorkflow.node_port)
            //处理Hosts及httppath，跟后端处理相同，将数据转成map[host]=httpPaths的格式
            if (this.createWorkflow.type == 'Ingress') {
                let hosts = new Map()
                let httpPaths = []
                let httpPath = {
                    path: this.createWorkflow.path,
                    path_type: this.createWorkflow.path_type,
                    service_name: this.createWorkflow.name,
                    service_port: parseInt(this.createWorkflow.port)
                }
                httpPaths.push(httpPath)
                hosts[this.createWorkflow.host] = httpPaths
                this.createWorkflowData.params.hosts = hosts
            }
            //发送请求
            httpClient.post(this.createWorkflowData.url,
this.createWorkflowData.params)
            .then(res => {
                this.$message.success({
                message: res.msg
                })
                this.getWorkflows()
            })
```

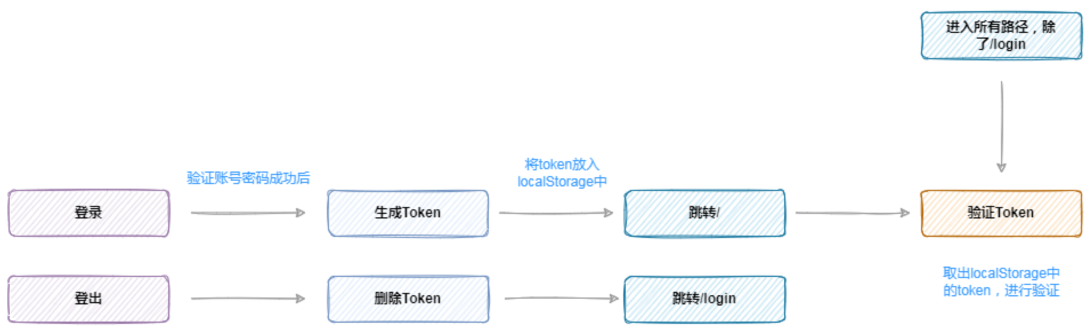
```
              .catch(res => {
                  this.$message.error({
                  message: res.msg
                  })
              })
              this.resetForm('createWorkflow')
              this.createWorkflowDrawerIndex2_2 = false
              this.active = 3
              this.fullscreenLoading = false
          },
          resetForm(formName) {
              this.$refs[formName].resetFields()
          }
      }
  }
</script>
```

（5）数据表格

workflow信息

```
<el-table-column width="20"></el-table-column>
<el-table-column min-width="50" align=left label="ID" prop="id"></el-table-column>
<el-table-column min-width="100" label="Workflow名">
    <template v-slot="scope">
        <a class="workflow-body-workflowname">{{ scope.row.name }}</a>
    </template>
</el-table-column>
<el-table-column label="类型" prop="type">
    <template v-slot="scope">
        <el-tag type="warning">{{ scope.row.type }}</el-tag>
    </template>
</el-table-column>
<el-table-column label="实例数" prop="replicas"></el-table-column>
<el-table-column min-width="100" label="deployment" prop="deployment"></el-table-column>
<el-table-column min-width="150" label="service" prop="service"></el-table-column>
<el-table-column min-width="150" label="ingress" prop="ingress"></el-table-column>
<el-table-column align=center min-width="150" label="创建时间">
    <template v-slot="scope">
        <el-tag type="info">{{ timeTransNot8(scope.row.created_at) }} </el-tag>
    </template>
</el-table-column>

<script>
export default {
    methods: {
        timeTransNot8(timestamp) {
            let date = new Date(new Date(timestamp).getTime() + 8 * 3600 * 1000)
            date = date.toJSON();
            date = date.substring(0, 19).replace('T', ' ')
            return date
        }
    }
}
</script>
```

8、登录/登出



(1) 登录

```
<template>
    <div class="login">
        <!-- 用户登录卡片 -->
        <el-card class="login-card">
            <template #header>
                <div class="login-card-header">
                    <span>用户登录</span>
                </div>
            </template>
            <!-- 表单 -->
            <el-form :model="loginData" :rules="loginDataRules" ref="loginData">
                <el-form-item prop="username">
                    <!-- 用户名 -->
                    <el-input prefix-icon="UserFilled" v-
model.trim="loginData.username" maxlength="32" placeholder="请输入账号" clearable></el-
input>
                </el-form-item>
                <el-form-item prop="password">
                    <!-- 密码 -->
                    <el-input prefix-icon="Lock" v-model.trim="loginData.password"
maxlength="16" show-password placeholder="请输入密码" clearable></el-input>
                </el-form-item>
                <el-form-item>
                    <!-- 登录按钮 -->
                    <el-button type="primary" style="width: 100%;border-radius: 2px"
:loading="loginLoading" @click="handleLogin">登 录</el-button>
                </el-form-item>
            </el-form>
        </el-card>
    </div>
</template>

<script>
import common from "../common/Config";
import httpClient from '../../utils/request';
import moment from 'moment';
import jwt from 'jsonwebtoken';
export default{
    data() {
        return {
            //加载等待动画
            loginLoading: false,
            //登录验证的后端接口
            loginUrl: common.loginAuth,
            loginData: {
                username: '',
                password: ''
            },
            //校验规则
            loginDataRules: {
                username: [{
                    required: true,
                    message: '请填写用户名',
                    trigger: 'change'
                }],
                password: [{
                    required: true,
                    message: '请填写密码',
                    trigger: 'change'
                }],
            }
        }
    },
    methods: {
        //登录方法
        handleLogin() {
            httpClient.post(this.loginUrl, this.loginData)
            .then(res => {
                //账号密码校验成功后的一系列操作
                localStorage.setItem('username', this.loginData.username);
                localStorage.setItem('loginDate', moment().format('YYYY-MM-DD
HH:mm:ss'));
                //生成token
                let token = jwt.sign(this.loginData, 'adoodevops', { expiresIn: '10h'
});
                localStorage.setItem('token', token);
                //跳转至根路径
                this.$router.push('/');
                this.$message.success({
                    message: res.msg
                })
            })
            .catch(res => {
                this.$message.error({
                message: res.msg
                })
            })
        }
    }
}
</script>

<style scoped>
    .login {
        position: absolute;
        width: 100%;
        height: 100%;
        background: aquamarine;
        background-image: url(../../assets/img/login3.webp);
        background-size: 100%;
    }
    .login-card {
        position: absolute;
        left: 40%;
        top: 30%;
        width: 350px;
        border-radius: 5px;
        background: rgb(255, 255, 255);
        overflow: hidden;
    }
    .login-card-header {
        text-align: center;
    }
</style>
```

（2）JWT校验

router/index.js

```
//使用钩子函数对路由进行权限跳转
router.beforeEach((to, from, next) => {
    //验证jwt token是否合法
    jwt.verify(localStorage.getItem('token'), 'adoodevops', function (err) {
        //如果去的路径是/login，直接放行，不需要验证
        if (to.path === '/login') {
            next()
        //如果验证异常，则跳转到/login
        } else if (err) {
            next('/login');
        //如果token合法，则放行
        } else {
            next();
        }
    });
});
```

（3）登出

六、部署前后端代码

1、前端

（1）进入k8s-demo-fe项目根目录

（2）删除/node_modules

（3）执行 npm install

（4）运行 npm run serve

（5）浏览器打开 localhost:8080

（6）默认登录账号密码 admin 123456

2、后端

（1）要求golang版本1.13及以上

（2）进入k8s-demo项目根目录

（3）执行 go mod tidy

（4）运行 go run main.go

（5）测试接口响应 curl --location --request GET --X GET ' http://0.0.0.0:9090/api/k8s/pods?namespace=kube-system '

PS：由于启动了jwt验证，请求后端接口时需要携带Authorization头，故直接请求后端地址会报错。

解决方式：打开main.go文件，注销第21行

r.Use(middle.JWTAuth())

六、总结

到这里，整个项目的前端页面就开发完成了，完全掌握后会发现，开发前端页面也就是固定的几个流程，布局->小视图->axios请求。好了，开启你的运维开发之路吧！

> 阿良教育： **www.aliangedu.cn**