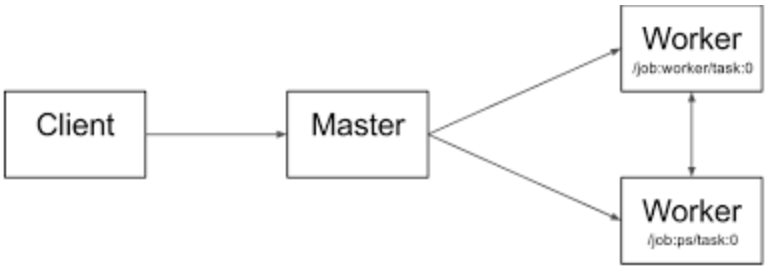
mini kuber

علیز که دیگر توانایی هندل کردن ددلاین‌های دانشگاهش را ندارد برای تنوع تصمیم گرفته است یک سیستم ساده‌شده‌ی scheduling بر پایه kubernetes بنویسد. اما چون حال این کار را هم ندارد تصمیم گرفته است این پروژه را به عنوان تمرین ap به ورودی‌ها بدهد.

مقدمه

در این تمرین شما باید یک mini kuber پیاده‌سازی کنید که در آن یک کلاینت به سرور master متصل می‌شود و درخواست انجام یک تسک را می‌دهد. سرور master نیز با توجه به ظرفیت هر کدام از node های worker، آن تسک را برای انجام به آن node می‌سپارد. توجه کنید که انجام تسک‌ها و گزارش نتیجه آنها هدف این تمرین نیست و تنها هدف این تمرین scheduling تسک‌ها است.

ساختار کلی کلاستر و نحوه ارتباط هر کدام از سرویس‌ها با یکدیگر را در عکس زیر می‌توانید مشاهده کنید.



توضیحات

سرور master

این سرور وظیفه مدیریت کلاستر را بر عهده دارد. یعنی تسک‌هایی که نیاز به انجام شدن دارند را کاربر روی این سرور ثبت می‌کند و سرور با توجه به nodeهای worker که در اختیار دارد تصمیم می‌گیرد تسک ثبت شده را در کدام نود schedule کند. همچنین توجه کنید که سرور مستر باید یک لیست از تمام nodeهای worker و یک لیست از اینکه هر کدام از تسک‌ها در چه وضعیتی و یا بر روی چه workerای هستند داشته باشد. وضعیت تسک‌ها نیز می‌تواند یکی از ۲ حالت running و pending باشد. در واقع سرور master اگر

نتواند تسکی را schedule کند آن را در حالت pending نگه می‌دارد تا در شرایط مناسب آن را schedule کند.

نودهای worker

نودهای ورکر وظیفه اجرا کردن تسک‌ها را دارند و سرور مستر تسک‌ها را بر روی nodeهای ورکر schedule می‌کند. هر نود ورکر ظرفیت محدودی دارد و تنها تعداد محدودی از تسک‌ها می‌توانند بر روی آن schedule شوند. از طرفی هر کدام از nodeهای worker یک id مشخص دارند که سرور master این id را برای هر worker انتخاب می‌کند. همچنین توجه کنید که nodeهای worker تنها از سرور مستر قابل دسترسی اند و client نمی‌تواند به آنها درخواست بفرستد.

کلاینت

کلاینت نیز تنها به سرور مستر متصل می‌شود و درخواست‌های خود مبنی برا ایجاد، حذف و یا مشاهده تسک‌ها را برای سرور مستر می‌فرستد و سرور مستر نیز با توجه به وضعیت nodeهای worker پاسخ مناسب را برای کلاینت ارسال می‌کند.

راه‌اندازی

ابتدا سرور master اجرا می‌شود و بر روی یک port در حال listen باقی می‌ماند و منتظر دریافت درخواست‌ها می‌شود. حال برنامه‌های worker اجرا می‌شوند و با ارسال یک درخواست به master خود را به عنوان ورکر معرفی می‌کنند و در صورت موفق بودن این فرآیند یک نود worker به لیست workerهای کلاستر اضافه می‌شود. توجه کنید که پورت سرور master به عنوان متغیر در برنامه‌ی ورکر ذخیره شده است.

همچنین یک متغیر دیگر در برنامه‌ی worker با نام MAX_TASK_NUMBER باید وجود داشته باشد که نشان‌دهنده‌ی حداکثر تعداد تسک‌هایی است که بر روی ورکر می‌توانند schedule شوند. پس از آنها کلاینت می‌تواند درخواست‌های خود را برای سرور master ارسال کند و سرور master نیز پاسخ‌های مناسب را باید برای کلاینت ارسال کند.

دستورات

برنامه‌ی شما باید قادر باشد که وقتی دستورات زیر در کلاینت وارد می‌شود آنها را بررسی کنید و پس از ارسال به سرور master پاسخ مناسبی را برای کاربر نمایش دهد.

ایجاد تسک جدید

```
k create task --name=task1
```

در صورت وارد شدن این دستور در کلاینت سرور مستر باید یک تسک با نام task1 در کلاستر ایجاد کند(مهم نیست این تسک بر روی چه ورکری قرار گیرد.) اگر ورکرها فضای مناسب و کافی داشتند تسک ایجاد می‌شود و یک پیام حاوی موفقیت آمیز بودن scheduling به همراه workerای که تسک بر روی آن schedule شده است برای کلاینت نمایش داده می‌شود. ولی اگر فضای مناسب در کل کلاستر وجود نداشت master باید این تسک را در حالت pending قرار دهد و این را به کاربر اعلام کند. از طرفی سرور master باید در زمانی که فضای کافی در کلاستر به وجود آمد آن تسک را schedule کند. در واقع در سرور مستر یک صف از تسک‌های در حالت انتظار وجود دارد که master در صورت به وجود آمدن فضای خالی(که با حذف شدن یک تسک ایجاد می‌شود) یک تسک با سیاست FIFO از این صف انتخاب می‌کند و schedule می‌کند.

همچنین وقتی بر روی یک worker یک تسک جدید schedule می‌شود باید یک لاگ در stdout آن worker نمایش داده شود که چه تسکی در چه زمانی بر روی این ورکر schedule شده است.

در صورتی که تسکی با این نام در کلاستر وجود داشت master باید پیغام مناسبی به کلاینت نمایش دهد.

```
k create task --name=task1 --node=worker1
```

این دستور همانند دستور بالاست با این تفاوت که task1 حتما باید بر روی نود با id=worker1 اجرا شود.

در صورتی که تسکی با این نام در کلاستر وجود داشت و یا workerای با نام مشخص شده در کلاستر وجود نداشت master باید پیغام مناسبی به کلاینت نمایش دهد.

نمایش تسک‌ها

```
k get tasks
```

این دستور یک لیست از تمام تسک‌های موجود در کلاستر شامل نام، وضعیت آنها (running, pending) و در صورتی که در حالت running بودند، workerای که تسک بر روی آنها schedule شده است به کاربر نمایش می‌دهد.

در صورتی که هیچ تسکی در کلاستر schedule نشده بود باید پیغام مناسبی به کاربر نمایش داده شود.

نمایش workerها

```
k get nodes
```

این دستور یک لیست از تمام نودهای worker شامل id و آدرس آنها(شامل ip , port) نمایش می‌دهد.

در صورتی که هیچ workerای در کلاستر وجود نداشت باید پیغام مناسبی به کاربر نمایش داده شود.

حذف کردن تسک‌ها

```
k create delete task --name=task1
```

این دستور تسک با نام task1 را از کلاستر حذف می‌کند.

در صورتی که همچین تسکی در کلاستر وجود نداشت باید پیغام مناسب نمایش داده شود.

توجه: در صورتی که کاربر دستوری وارد کند که با فرمت‌های ذکر شده تطابق نداشت عبارت زیر برای کاربر نمایش داده شود.

```
wrong command!
```

بخش امتیازی

توجه: انجام بخش‌های زیر نمره اضافه علاوه بر نمره اصلی تمرین خواهد داشت.

فعال و غیر فعال کردن worker

```
k cordon node [nodename]
```

این دستور یک نود worker را در حالت غیرفعال قرار می‌دهد. در این حالت دیگر تسکی نمی‌تواند روی این نود schedule شود و تسک‌هایی که از قبل روی این نود schedule شده بودند نیز باید دوباره عملیات scheduling بر روی آنها انجام شود. توجه کنید که در صورتی که از قبل تسک‌هایی بر روی این نود schedule شده باشند باید ابتدا یک لیست از این تسک‌ها در خروجی برای کلاینت نمایش داده شود. در هر صورت نی پیغام مناسبی مبنی بر drain شدن نود به کاربر نمایش داده شود.

همچنین در صورت انجام این بخش باید خروجی دستور نمایش نودها نیز عوض شود و وضعیت هر نود نیز در لیست قابل مشاهده باشد.(حالت نود در این صورت unschedulable است.)

اگر نودی با این اسم وجود نداشت یا در حالت فعلی نیز در حالت unschedulable باشد پیغام مناسب در کلاینت نمایش داده شود.

```
k uncordon node [nodename]
```

این دستور یک نود را از حالت غیرفعال خارج می‌کند و دوباره در حالت فعال قرار می‌دهد. بعد از اجرای این دستور تسک‌ها می‌توانند دوباره بر روی نود مورد نظر schedule شوند.

در صورتی که نودی با این اسم وجود نداشت و یا نود در حللت غیر فعال نبود پیغام مناسب چاپ شود.

load balancing

در این بخش شما باید سرور master را طوری طراحی کنید که پخش کردن تسک‌ها بر روی نودها را به صورت متوازن باشد. به طور دقیق‌تر تسک‌ها بر اساس درصد ریسورس باقی مانده در نودهای worker schedule شوند. به عنوان مثال اگر یک نود worker با MAX_TASK_NUMBER=2 و یک نود دیگر با

MAX_TASK_NUMBER=4 در کلاستر وجود داشته باشد و بر روی هر دو نود ۱ تسک schedule شده باشد نود اول 50% ریسورس آزاد دارد و نود دوم 75%. بنابراین تسک جدید بر روی نود دوم schedule می‌شود.

keep-alive

در این بخش شما باید مکانیزمی طراحی کنید که در صورتی که یکی از نودهای worker از شبکه خارج شد master این مسئله را متوجه شود و تسک‌هایی که روی آن وجود داشتند را بر روی نود دیگری schedule کند و همچنین state این نود به حالت NetworkUnavailable تغییر پیدا خواهد کرد و master دیگر نباید بر روی آن تسکی schedule کند.

برای طراحی این مکانیزم می‌تواند از دو روش پیشنهادی زیر استفاده کنید.

۱. نودهای worker به صورت متناوب یک بسته برای master ارسال کنند و در صورتی که master بعد از یک مدت زمان معین بسته‌ای از سمت ورکر دریافت نکرد اینگونه فرض کند که worker از شبکه خارج شده است.

۲. نود master به صورت متناوب یک بسته برای workerها ارسال کند و در صورتی که پس از چند بار از یک worker پاسخی دریافت نکرد اینگونه فرض کند که آن worker از شبکه خارج شده است.