MioBook

مقدمه

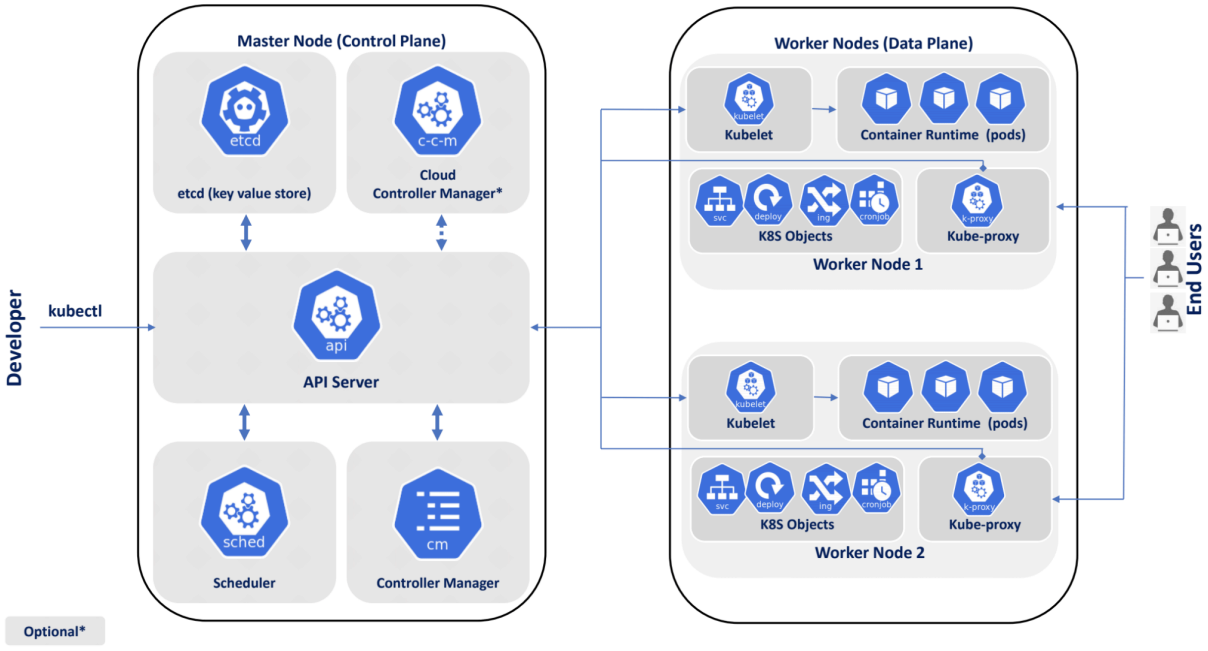
هدف از این پروژه، آشنایی با ابزار کوبرنتیز (Kubernetes) است. شما در پروژه قبلی بک‌اند و فرانت‌اند خود را Dockerize کردید و با استفاده از ابزار Docker Compose آنها را به همراه پایگاه‌داده به هم متصل کردید.
ابزار کوبر نیز همانند Docker Compose یک ابزار Container Orchestration است که برخلاف Compose که همه کانتینرها را بر روی یک Node بالا می‌آورد، توزیع شده است و قابلیت‌های بسیار زیادی دارد که شما با برخی از آنها آشنا خواهید شد.

ابزار Kubernetes

شما با مفاهیم اولیه کوبر در کلاس‌های درس آشنا شده‌اید.
اسلایدهای درس را یک دور مرور کنید و با کلاستر کوبر بیشتر آشنا شوید.
جالب است بدانید که چون بین حرف اول و آخر کوبر 8 کاراکتر قرار دارد، به آن k8s نیز می‌گویند.

معماری کوبر

یک شمای کلی از معماری کوبر در شکل زیر آورده شده است:



از آنجا که کوبر توزیع‌شده است، هر node یا سیستم فیزیکی می‌تواند در نقش control plane و یا worker باشد. معمولا برای HA (High Availability)، تعداد control plane-ها بیشتر از یکی بوده تا پایگاه‌داده etcd (که یک پایگاه‌داده NoSQL و key-value است) به طور توزیع شده کار کرده و با وجود replica-ها، با پایین آمدن یک node، کلاستر دچار مشکل نشود. تعداد پیشنهاد شده برای کلاسترها 3 یا 5 است. پایگاه‌داده etcd برای Leader Election بین node-های خود به تعداد فرد تا node نیاز دارد.
درون worker node-ها، آبجکت‌های مختلف کوبر که جلوتر آنها را می‌بینیم (مانند deployment) ساخته می‌شوند. ارتباط control plane با worker node-ها توسط Kubelet در آنها صورت می‌گیرد.

کلاستر کوبر

هدف این تمرین این است که کانتینرهای بک‌اند، فرانت‌اند و پایگاه‌داده خود را روی یک کلاستر کوبر، به شکل توزیع‌شده بالا بیاوریم.
برای این کار به یک کلاستر (یک مجموعه node که برخی control plane و برخی worker node اند) نیاز داریم. معمولا کلاسترهای واقعی روی چندین سرور متفاوت با استفاده از ابزار kubeadm راه‌اندازی می‌شوند.
برای کارهای آزمایشی، می‌توانیم با استفاده از ابزارهایی مانند Minikube، Kind و یا K3s یک کلاستر به صورت Local در سیستم خود بالا بیاوریم. این ابزارها هم برای یادگیری و تست کردن چیزها روی کوبر مناسب اند و هم برای اجرای integration test-ها (که باید ارتباط بین چند سرویس را تست کرد) استفاده می‌شوند.

ابزار Kubectl

ما پس از راه‌اندازی کلاستر کوبر، باید با ابزار kubectl که ابزار کامندلاین رسمی اتصال به API کوبر است به آن متصل شویم. بنابراین ابتدا kubectl را برای سیستم خود با استفاده از این لینک نصب کنید. از آنجا که هر نسخه از kubectl متناسب با نسخه کلاستر کوبر است، آخرین نسخه آن را دریافت کنید چرا که کلاستری که با Kind خواهیم ساخت نیز روی آخرین نسخه کوبر خواهد بود. دستور `kubectl version` را جهت صحت نصب و دیدن نسخه آن اجرا کنید.

ابزار Kind

در این تمرین از ابزار Kind استفاده می‌کنیم. Kind مخفف Kubernetes in Docker است و همانطور که از نامش مشخص است، از کانتینرهای داکر به عنوان node-های کلاستر کوبر استفاده می‌کند. برای مقایسه، ابزار Minikube یک VM واقعی بالا آورده تا در آن کلاستر راه‌اندازی شود.
این ابزار را از این لینک نصب کنید. پس از نصب، با استفاده از آن یک کلاستر لوکال بسازید. توجه کنید که تعداد node-ها و نقش آنها پس از ساختن کلاستر در Kind قابل تغییر نیستند. پس در صورتی که با استفاده از دستور `kind create cluster` یک کلاستر جدید ساخته باشید، از آنجا که صرفا یک node در نقش control plane دارد، کاری نمی‌توان روی آن کرد. بنابراین مطابق لینک داده شده، یک کانفیگ Cluster بنویسید و با دستور:

```
kind create cluster --name sample --config cluster-config.yaml
```
یک کلاستر با سه worker node بسازید.

فایل Kubeconfig

برای اینکه بتوانیم با ابزار kubectl به کلاستر مد نظر خود وصل شویم، باید در فایل kubeconfig کلاستر را به kubectl معرفی کنیم. این فایل در محل زیر قرار دارد:

```
~/.kube/config  |  C:\Users\Username\.kube\config
```

با اجرای دستور ساخت کلاستر در Kind، این فایل به طور خودکار آپدیت شده و مشخصات کلاستر جدید در آن ثبت می‌شود و بدون نیاز به تغییری، kubectl به آن متصل می‌شود.

برای اینکه بتوان با چندین کلاستر کار کرد، kubectl مفهومی به نام context دارد که در آن فایل هم مشاهده می‌کنید. هر کانتکست ترکیبی از کلاستر مد نظر و یوزر آن، و چیزهای دیگر مانند namespace است. می‌توانید با زیرمجموعه‌های کامند `kubectl config current-context/get-contexts/use-context` آنها را ببینید.

حال می‌توانید با دستور `kubectl get nodes` وضعیت node-های کلاستر را مشاهده کنید.

آبجکت‌های کوبر

احتمالا تا کنون کمی YAML کوبر دیده‌اید. YAML یک فرمت خوانا برای تبادل داده یا نوشتن کانفیگ‌ها همانند JSON است. اگر به کانفیگ ساخت کلاستر Kind و یا فایل Kubeconfig دقت کنید، می‌بینید که فیلدهای apiVersion و kind در آنها وجود دارد.

kind یک کانفیگ، یک resource را مشخص می‌کند. مثلا می‌توانیم kind: Pod داشته باشیم که یعنی داریم اینستنس از یک resource کوبر به نام pod می‌سازیم. خود اینستنس یک آبجکت کوبر محسوب می‌شود. یکی از قابلیت‌های قدرتمند کوبر CRD یا Custom Resource Definition است که اجازه می‌دهد علاوه بر انواع resource-های خود کوبر، resource-های خودمان را نیز تعریف کنیم و یا از resource-های ساخته شده توسط دیگران استفاده کنیم. در این تمرین ما به اینها نمی‌پردازیم و فقط برخی آبجکت‌های خود کوبر را می‌سازیم.

برای اعمال کردن هر YAML کوبر، کافیست که دستور `kubectl apply -f file.yaml` را وارد کنیم. همچنین می‌توانیم YAML داخل کلاستر را با `kubectl edit` تغییر دهیم.

Pod در کوبر

یک Pod در کوبر کوچک‌ترین واحد اجرایی است. داخل هر Pod ممکن است یک یا چند کانتینر قرار بگیرد. این یعنی از روی image-های داکر، کانتینر ساخته شده و در Pod قرار می‌گیرد. این کانتینرها IP و فضا و... مشترک خواهند داشت. در واقع Pod یک مفهوم بالای کانتینر است. معمولا در هر Pod یک کانتینر بیشتر قرار نمی‌دهند.

برای اینکه چندین Pod از یک image داشته باشیم، از ReplicaSet و یا آبجکت‌ای انتزاعی‌تر به نام Deployment استفاده می‌کنیم. این لینک را جهت آشنایی با بخش‌های کوبر مطالعه کنید.

بک‌اند

سرویس بک‌اند یک سرویس Stateless است. ما می‌توانیم چندین اینستنس از بک‌اند خود را داشته باشیم و به طور مثال ریکوئست کاربر را به یکی از آنها Load Balance کنیم. این یعنی بک‌اند قابلیت Horizontal Scaling را به راحتی دارد و فقط کافیست که تعداد آنها را افزایش دهیم.

شما در این بخش باید یک **Deployment** کوبر برای سرویس بک‌اند بنویسید. حتما در این باره جست‌وجو کنید و فیلدهای مختلف یک Deployment را ببینید. پس از apply کردن آبجکت روی کلاستر، می‌توانید با `kubectl get deployments` آن را دیده و با `kubectl get pods` پادهای آن را مشاهده کنید.

در deployment خود، تعداد replica-های بک‌اند را 2 عدد در نظر بگیرید و از image-ای که فاز قبل برای بک‌اند ساختید استفاده کنید. در اینجا می‌بینید که شباهت‌هایی با Docker Compose-ای که نوشته اید وجود دارد. مثلا در اینجا نیز باید environment variable-ها را به کانتینر آن بدهید و یا health-check بنویسید و رمز پایگاه‌داده را به صورت secret بدهید.

برای پادهای خود محدودیت روی Memory و CPU نیز قرار دهید و با واحد میلی‌کور آشنا شوید.

برای دادن env-ها از یک آبجکت **ConfigMap** کوبر استفاده کنید. این آبجکت ساده کانفینگ‌های شما را روی یک آبجکت در کلاستر ذخیره می‌کند و شما می‌توانید در deployment خود مقادیر env را از روی آنها بدهید.

برای رمز پایگاه‌داده، از یک آبجکت **Secret** کوبر استفاده کنید. این آبجکت مشابه ConfigMap بوده ولی در واقع خود YAML آن محرمانه نگه داشته می‌شود و هر کسی نمی‌تواند مقدار داخل آبجکت آن را از روی کلاستر بخواند.

حال برای دادن رمز به بک‌اند، یا می‌توانید همانند ConfigMap مقدار Secret را صاف به environment variable متناظر آن تنظیم کنید، و یا همانند Docker Compose آن را داخل یک فایل درون کانتینر قرار دهید. برای راه دوم که کمی امن‌تر ولی پیچیده‌تر است (فرقی ندارد که شما از کدام روش استفاده می‌کنید)، باید یک volume تعریف کنید که در آن از روی مقادیر Secret فایل‌ها ساخته می‌شوند.

در نهایت، به یک آبجکت **Service** نیاز داریم. Service در کوبر برای ارتباط بقیه اجزا با یک سرویس تعریف می‌شود. Service انواع مختلفی دارد که برای بک‌اند از نوع پیش‌فرض ClusterIP استفاده می‌کنیم. این به این معناست که سرویس ما فقط در داخل کلاستر قابل دسترسی است و از بیرون شبکه دیده نمی‌شود. هدف Service این است که با یک hostname یا IP ثابت بتوانیم به یک پاد از پادهای deployment بک‌اند که ممکن است هر لحظه دچار تغییر شوند (مثلا از یک node به یک node دیگر بروند) متصل شویم.

شما می‌توانید با `kubectl logs -f pod-name` که اسم پاد را از get pods می‌گیرید، لاگ‌های کانتینر خود را مشاهده کنید. همچنین با استفاده از `kubectl exec -it pod-name -- bash` می‌توانید شبیه SSH وارد یک پاد بشوید.

برای اینکه بتوانید از بیرون برای تست به پاد بک‌اند خود ریکوئست بزنید، می‌توانید از قابلیت port-forward کوبر استفاده کنید. دستور `kubectl port-forward pod-name 80:1234` پورت 1234 داخل پاد را به طور موقت روی پورت 80 در کامپیوتر شما قرار می‌دهد.

پایگاه‌داده

از آنجا که پایگاه‌داده سرویس Stateful است، صرفا به یک پاد از آن نیاز داریم و همچنین انتظار داریم که داده‌های آن با ریستارت شدن پاد تغییر نکند. بنابراین همانند Docker Compose، در اینجا نیز باید یک Volume ثابت برای آن در نظر بگیریم.

Volume در کوبر به این صورت است که باید ابتدا آبجکت PV یا **PersistentVolume** وجود داشته باشد و نشان دهد که مقداری فضا برای استفاده توسط دیگران وجود دارد. سپس پاد ما با استفاده از یک PVC یا **PersistentVolumeClaim** می‌تواند درخواست مقدار مشخصی از دیسک کند.

در خیلی از کلاسترهای واقعی، نیازی به ساختن PV نیست و صرفا با ساختن PVC به طور خودکار فضای مورد نیاز اختصاص می‌یابد. به این پروسه اتومات Dynamic Provisioning می‌گویند. در مقابل آن، با ساخت PV داریم Static Provisioning انجام می‌دهیم.

می‌توانید در صورت تمایل، از این ابزار برای داشتن dynamic provisioning در Kind استفاده کنید.

بنابراین برای پایگاه‌داده نیز یک **ConfigMap** و **Deployment** با یک replica تعریف کنید که در آن از volume گرفته شده استفاده می‌شود. همچنین یک **Service** از نوع ClusterIP همانند بک‌اند تعریف کنید.

برای رمز پایگاه‌داده از Secret ساخته شده استفاده کنید.

برای چنین سرویس‌هایی که stateless نیستند، می‌توانیم به جای deployment از **StatefulSet** استفاده کنیم. در مورد آن مطالعه کنید و اگر خواستید به جای Deployment از آن استفاده کنید.

فرانت‌اند

برای سرویس فرانت‌اند که نیز Stateless است، یک Deployment، ConfigMap و Service تعریف کنید.

در اینجا می‌خواهیم که فرانت‌اند از بیرون شبکه کلاستر قابل دسترسی باشد. برای این کار در واقع باید یک آبجکت Ingress تعریف کرد و دامنه و IP پابلیک داشت. برای همین در اینجا صرفا با استفاده از port-forward کردن به Service فرانت‌اند (`kubectl port-forward service/front-end-service-name 80:3000`) آن را در مرورگر خود باز می‌کنیم.

گزارش

پس از بالا آمدن و کار کردن سیستم MioBook، از خروجی `kubectl get pods` و صفحه یک کتاب در فرانت‌اند در کنار port-forward کردن سرویس عکس بگیرید. همچنین پاسخ سوالات زیر را نیز نوشته (به طور خلاصه) و در یک PDF در کنار پروژه خود قرار دهید.

1. انواع Service-ها در کوبر چه کاربرد و محدودیتی دارند؟
2. Namespace-ها در کوبر چیستند و چگونه namespace پیشفرض را تنظیم کنیم؟ (هم با kubectl و هم با ادیت kubeconfig)
3. در صورتی که لود بک‌اند زیاد بشود، چگونه با kubectl تعداد پادها را افزایش می‌دهیم؟ چگونه می‌توانیم این کار را خودکار انجام دهیم؟ (HPA)
4. با اعمال تغییرات روی یک آبجکت مثلا deployment، کوبر سعی می‌کند با اعمال تغییرات روی سیستم، حالت کنونی را از yaml قبلی به yaml جدید برساند. این پروسه توسط چه واحدهایی انجام می‌شود؟
5. Operator-ها در کوبر چیستند؟
6. تفاوت liveness، readiness و startup probe در کوبر چیست؟
7. تفاوت Deployment و ReplicaSet در کوبر چیست؟ DeamonSet برای چه کاری استفاده می‌شود؟
8. تفاوت requests و limits در بخش resources یک Deployment چیست؟

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود. برای تحویل آن کافی است که یکی از اعضای گروه، لینک مخزن گیت‌هاب و Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. پروژه شما بر روی این کامیت مورد ارزیابی قرار می‌گیرد.
- حتما کاربر IE-S04 را به پروژه خود اضافه کنید.
- ساختار مناسب و تمیزی کد برنامه، بخشی از نمره همه پروژه‌های شما خواهد بود. بنابراین در طراحی ساختار برنامه و همچنین خوانایی کد دقت زیادی به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده شباهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.