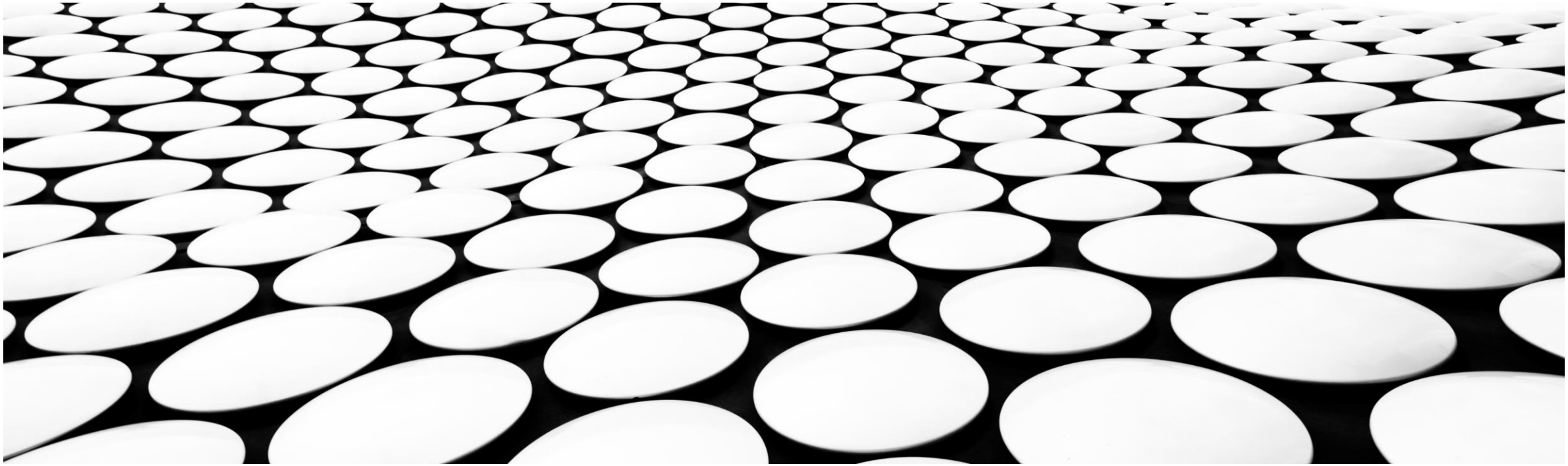
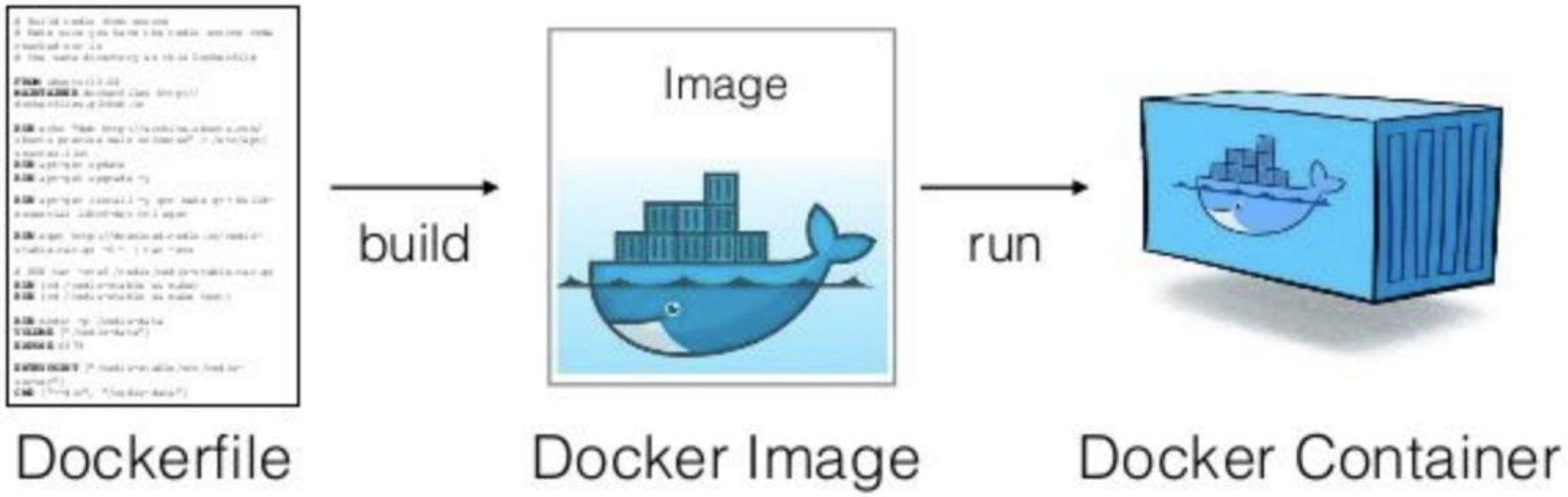


服务端开发-容器镜像构建与编排

陶召胜

由Dockerfile构建镜像



Dockerfile文件的指令

- FROM：指定基础镜像，必须为第一个命令
- RUN：构建镜像时执行的命令
- ADD：将本地文件添加到容器中，tar类型文件会自动解压
- COPY：功能类似ADD，但是不会自动解压文件
- CMD：构建容器后调用，也就是在容器启动时才进行调用
- ENTRYPOINT：配置容器，使其可执行化。配合CMD可省去“application”，只使用参数，用于docker run时根据不同参数执行不同功能
- LABEL：用于为镜像添加元数据
- ENV：设置环境变量
- EXPOSE：指定与外界交互的端口，容器内的端口号，docker run时加-P则会映射一个随机号（宿主机）
- VOLUME：用于指定持久化目录，docker run时如果没有指定挂载目录，会创建一个volume
- WORKDIR：工作目录，类似于cd命令
- USER：指定运行容器时的用户名或 UID
- ARG：用于指定传递给构建运行时的变量
- ONBUILD：用于设置镜像触发器

Dockerfile练习

- 目标：生成镜像，并运行容器
- 代码下载路径：https://github.com/tzs919/mysite.git
- 命令：
  - ✓ 生成镜像：docker build -t mysite:latest .
  - ✓ 运行容器：docker run -d --name mysite -p 8081:80 mysite:latest
- 访问：
  - ✓ http://localhost:8081/admin/ 用户名：admin，口令：12345678
  - ✓ http://localhost:8081/polls/

context

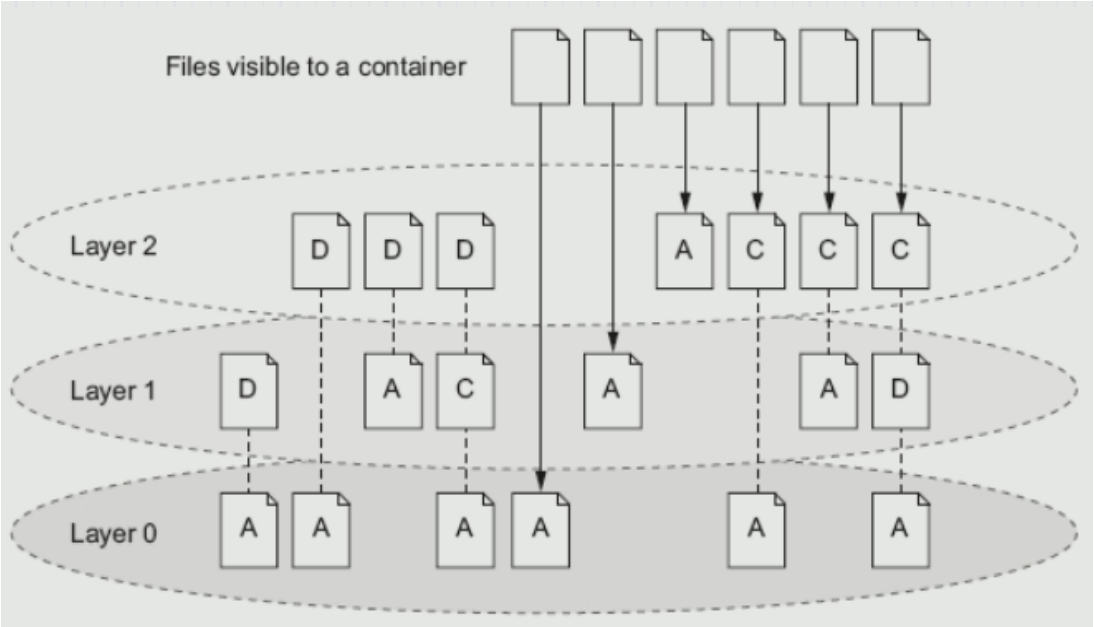
- 最后一个参数：.

Docker build

- docker build [OPTIONS] PATH | URL | -
- 如何编写最佳的Dockerfile： https://zhuanlan.zhihu.com/p/26904830
  - ✓ .dockerignore文件
  - ✓ 容器只运行单个应用
  - ✓ 将多个RUN指令合并为一个
  - ✓ 基础镜像的标签不要用latest
  - ✓ 每个RUN指令后删除多余文件
  - ✓ 选择合适的基础镜像(alpine版本最好)
  - ✓ 设置WORKDIR和 CMD
- Docker健康检查HEALTHCHECK的使用方法：https://zhuanlan.zhihu.com/p/386986915

镜像分层

- 写时复制(COW, Copy-On- Write)
- docker history <image name> 查看镜像的层



镜像分层--将所有的RUN指令合并为一个

- Dockerfile中的每个指令都会创建一个新的镜像层
- 镜像层将被缓存和复用
- 当Dockerfile的指令修改了，复制的文件变化了，或者构建镜像时指定的变量不同了，对应的镜像层缓存就会失效
- 某一层的镜像缓存失效之后，它之后的镜像层缓存都会失效
- 镜像层是不可变的，如果我们在某一层中添加一个文件，然后在下一层中删除它，则镜像中依然会包含该文件(只是这个文件在Docker容器中不可见了)

服务编排工具，docker-compose

- Compose 项目是 Docker 官方的开源项目，负责实现对 Docker 容器集群的快速编排
- 一个单独的 docker-compose.yml 模板文件（YAML 格式）来定义一组相关联的应用容器为一个项目（project）
- Compose 的默认管理对象是项目，通过子命令对项目中的一组容器进行便捷地生命周期管理
- Compose 中有两个重要的概念
  - ✓ 服务 (service)：一个应用的容器（可能会有多个容器），实际上可以包括若干运行相同镜像的容器实例
  - ✓ 项目 (project)：由一组关联的应用容器组成的一个完整业务单元，在 docker-compose.yml 文件中定义
- 使用微服务架构的系统一般包含若干个微服务，每个微服务一般部署多个实例。如果每个服务都要手动启停，那么效率低，维护量大

docker-compose.yml格式

- https://blog.csdn.net/weixin_46545831/article/details/112995427
- https://zhuanlan.zhihu.com/p/93459395

YAML文件

- 使用缩进表示层级关系，不允许使用Tab键，只允许使用空格
- # 表示注释，从这个字符一直到行尾，都会被解析器忽略。
- 对象，键值对，使用冒号结构表示
  - ✓ animal: pets
  - ✓ hash: { name: Steve, foo: bar }
- 数组,一组连词线开头的行，构成一个数组

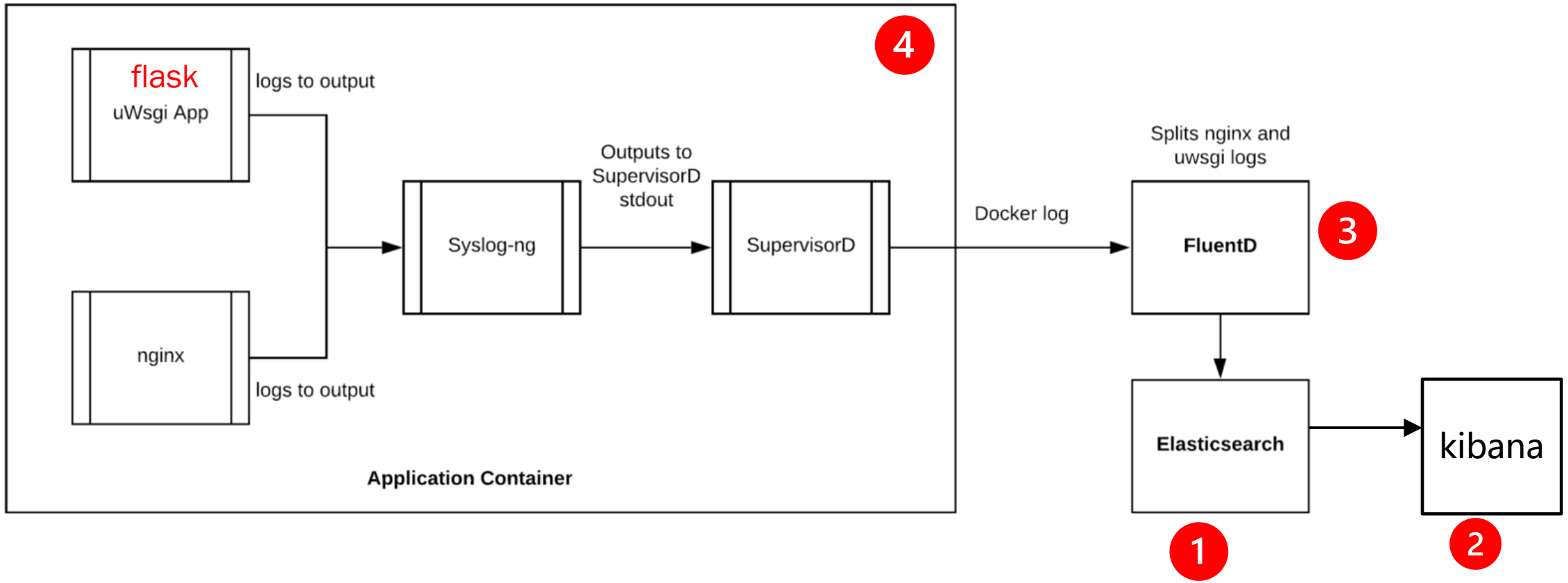
  - Cat
  - Dog
  - Goldfish

  - ✓ 行内表示法：animal: [Cat, Dog]

docker-compose常用命令

- docker-compose --help
- docker-compose up -d # 该命令十分强大，它将尝试自动完成包括构建镜像，（重新）创建服务，启动服务，并关联服务相关容器的一系列操作
- docker-compose ps、docker-compose ps --services
- docker-compose images
- docker-compose stop # 终止整个服务集合
- docker-compose stop nginx # 终止指定的服务 （这有个点就是启动的时候会先启动 depond_on 中的容器，关闭的时候不会影响到 depond_on 中的）
- docker-compose logs -f [services...] # 查看容器的输出日志
- docker-compose build [SERVICE...]
- docker-compose rm nginx # 移除指定的容器
- docker-compose up -d --scale flask=3 organizationservice=2 #设置指定服务运行的容器个数

docker-compose练习



ports、expose、links、depends_on

- https://zhuanlan.zhihu.com/p/382779508
- ports
  - ✓ 用来把服务端口映射给宿主机
- expose
  - ✓ 用来把服务端口开放给其他服务，客户端服务可以通过links功能访问服务端服务的端口
- links
  - ✓ links的目的是把一个服务的名称在当前服务里面创建一个别名
- depends_on
  - ✓ 当前服务启动之前先要把depends_on指定的服务启动起来才行

如何配置 Docker Logging 驱动

- Logging 驱动是可以插拔的框架
- Logging 驱动是 Container 用来使用服务来访问 log data 的工具
- Docker 支持很多种类的 Logging Driver
- 默认驱动（log-driver）："json-file"，存放到主机目录：/var/lib/docker/containers/[容器ID]-json.log
- 单个容器的日志配置
  - ✓ docker run --log-driver syslog nginx
  - ✓ docker run --log-driver json-file --log-opt max-size=50m nginx
- 更多驱动：fluentd、splunk、awslogs

启动顺序

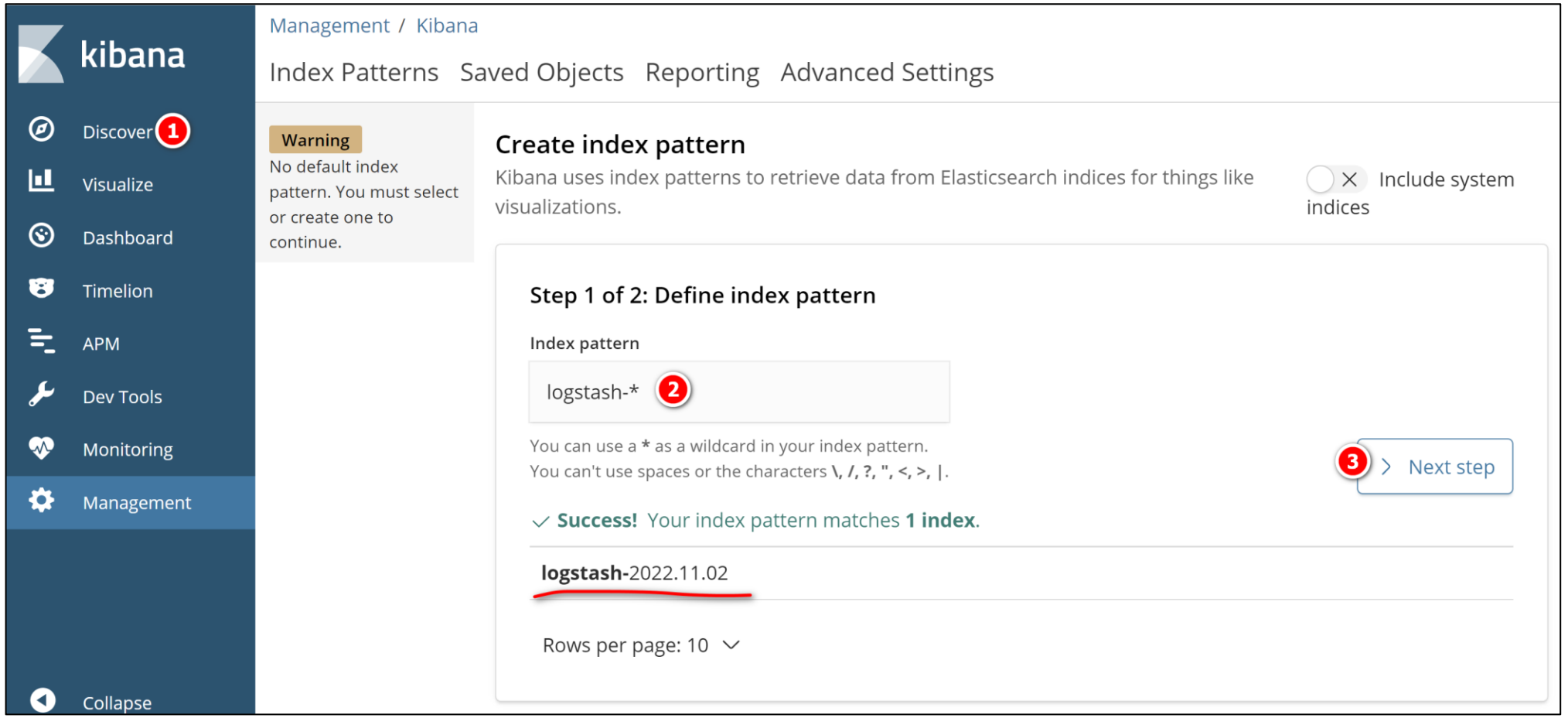
需要按以下顺序启动

1. docker-compose up -d elasticsearch kibana
   - ✓ 稍等约1分钟，等完全可用
2. docker-compose up -d fluentd
   - ✓ 稍等约半分钟，等完全可用
3. docker-compose up -d flask

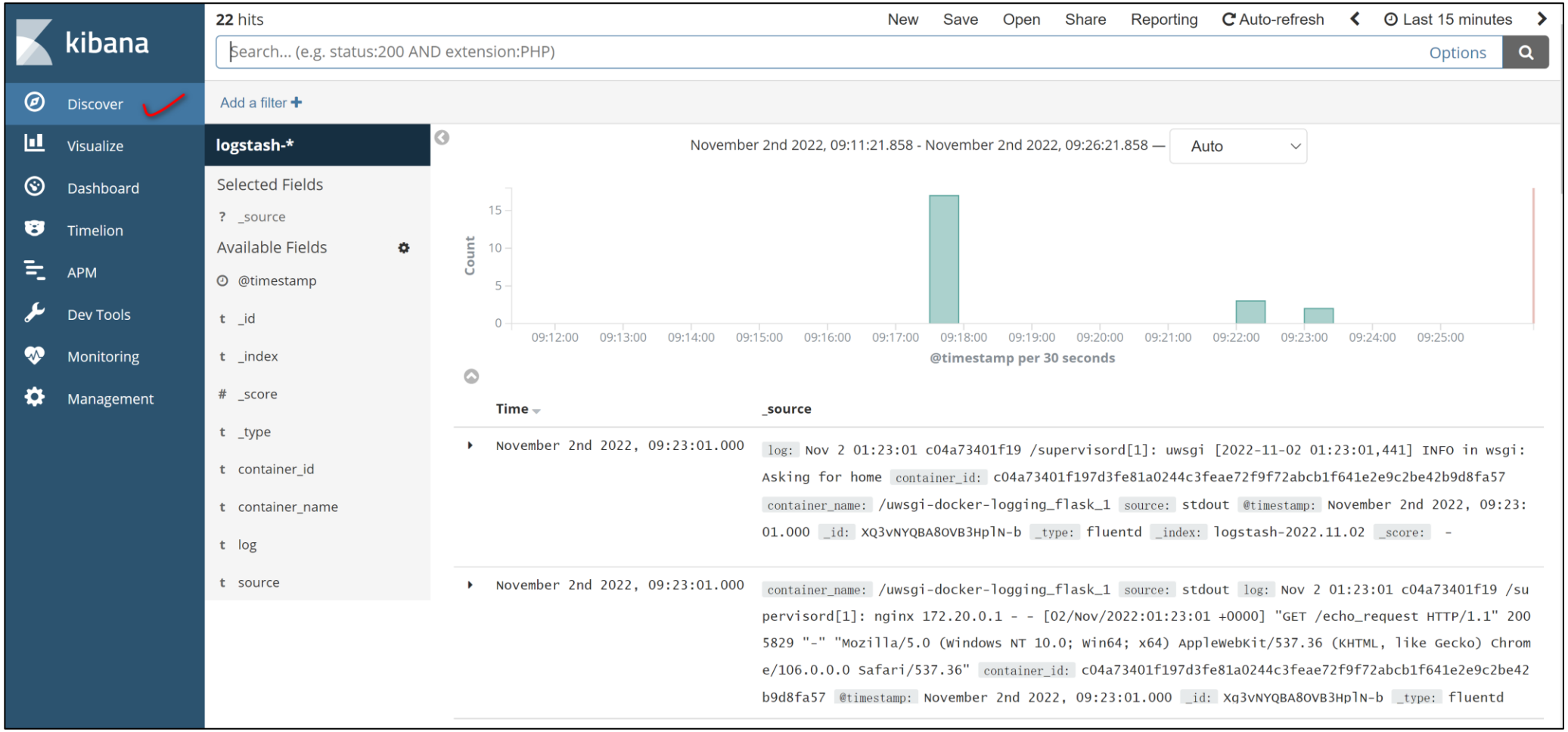
使用

- 日志搜索
  - ✓ kibana：http://localhost:5601
- web程序访问（flask程序）
  - ✓ http://localhost:8090/
  - ✓ http://localhost:8090/echo_request

Create index pattern



结果页面


kibana
22 hits
New
Save
Open
Share
Reporting
Auto-refresh
Last 15 minutes
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter +
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
logstash-*
Selected Fields
? _source
Available Fields
@timestamp
t _id
t _index
_score
t _type
t container_id
t container_name
t log
t source
November 2nd 2022, 09:11:21.858 - November 2nd 2022, 09:26:21.858 —
Auto
Count
15
10
5
0
09:12:00
09:13:00
09:14:00
09:15:00
09:16:00
09:17:00
09:18:00
09:19:00
09:20:00
09:21:00
09:22:00
09:23:00
09:24:00
09:25:00
@timestamp per 30 seconds
Time
_source
November 2nd 2022, 09:23:01.000
log: Nov 2 01:23:01 c04a73401f19 /supervisord[1]: uwsgi [2022-11-02 01:23:01,441] INFO in wsgi: Asking for home container_id: c04a73401f197d3fe81a0244c3feae72f9f72abcb1f641e2e9c2be42b9d8fa57 container_name: /uwsgi-docker-logging_flask_1 source: stdout @timestamp: November 2nd 2022, 09:23:01.000 _id: XQ3vNYQBA8OVB3HplN-b _type: fluentd _index: logstash-2022.11.02 _score: -
November 2nd 2022, 09:23:01.000
container_name: /uwsgi-docker-logging_flask_1 source: stdout log: Nov 2 01:23:01 c04a73401f19 /supervisord[1]: nginx 172.20.0.1 - - [02/Nov/2022:01:23:01 +0000] "GET /echo_request HTTP/1.1" 200 5829 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36" container_id: c04a73401f197d3fe81a0244c3feae72f9f72abcb1f641e2e9c2be42b9d8fa57 @timestamp: November 2nd 2022, 09:23:01.000 _id: Xq3vNYQBA8OVB3HplN-b _type: fluentd

docker-compose.yml格式参考

```
# 指定服务名称
webapp:
  # 指定服务使用的镜像
  image: aphysia/dockerdemo
  # 指定容器名称
  container_name: dockerdemo
  # 指定服务运行的端口
  ports:
    - 8081:8081
  # 指定容器中需要挂载的文件
  volumes:
    - /etc/localtime:/etc/localtime
    - /tmp/dockerdemo/logs:/var/logs
```

```
version: "3"
services:
  web1:
      image: nginx
      healthcheck:
        test: ["CMD", "curl", "-f", "http://localhost"]
        interval: 5s
        timeout: 3s
        retries: 3
```

```
services:
  redis:
    image: redis:latest
    restart: always
    ports:
      - "6389:6379"
    volumes:
      - /tmp/redis.conf:/etc/redis/redis.conf
    command: redis-server /etc/redis/redis.conf
```

```
version '3'
services:
   db:
     image:mysql:5.7
     volumes:
        - db_data:/var/lib/mysql
     restart: always
     environment:
        MYSQL_ROOT_PASSWORD:wordpress
        MYSQL_DATABASE:wordpress
        MYSQL_USER:wordpress
        MYSQL_PASSWORD:wordpress
   wordpress:
     depends_on:
      -db
      image:wordpress:latest
      ports:
        -"8081:80"
      restart:always
      environment:
         WORDPRESS_DB_HOST:db:3306
         WORDPRESS_DB_USER:wordpress
         WORDPRESS_DB_PASSWORD:wordpress
volumes:
  db_data:
```

下节课准备，K8S安装

- 请访问链接：https://github.com/AliyunContainerService/k8s-for-docker-desktop，根据说明安装

谢谢观看！