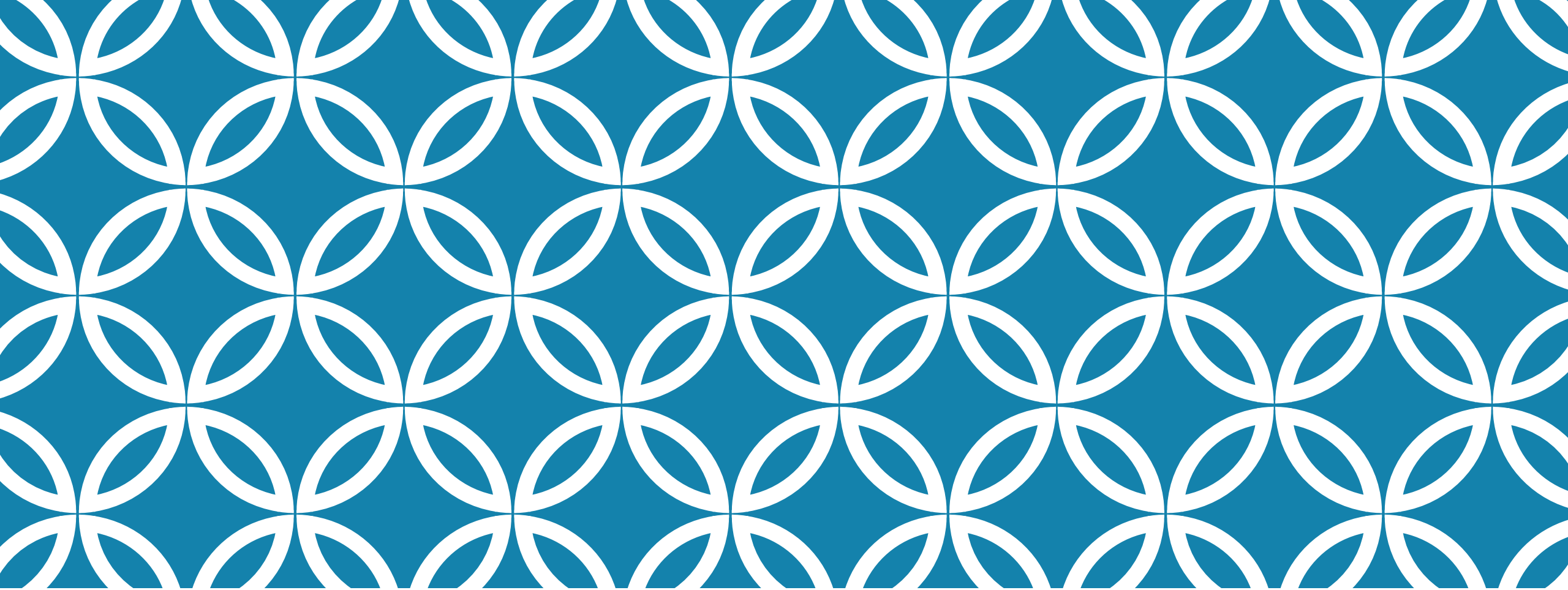


网易容器云平台容器化实践

云计算技术部平台架构组
朱凌墨

目次

本容器化实践特点

非胖容器

- 无其余「系统」进程开销，直接明了
- 容器启动时仅执行应用需要的进程及 PID 1 对策进程，且 PID 1 对策进程在启动其余进程后只等待信号处理，可认为没有影响性能的开销

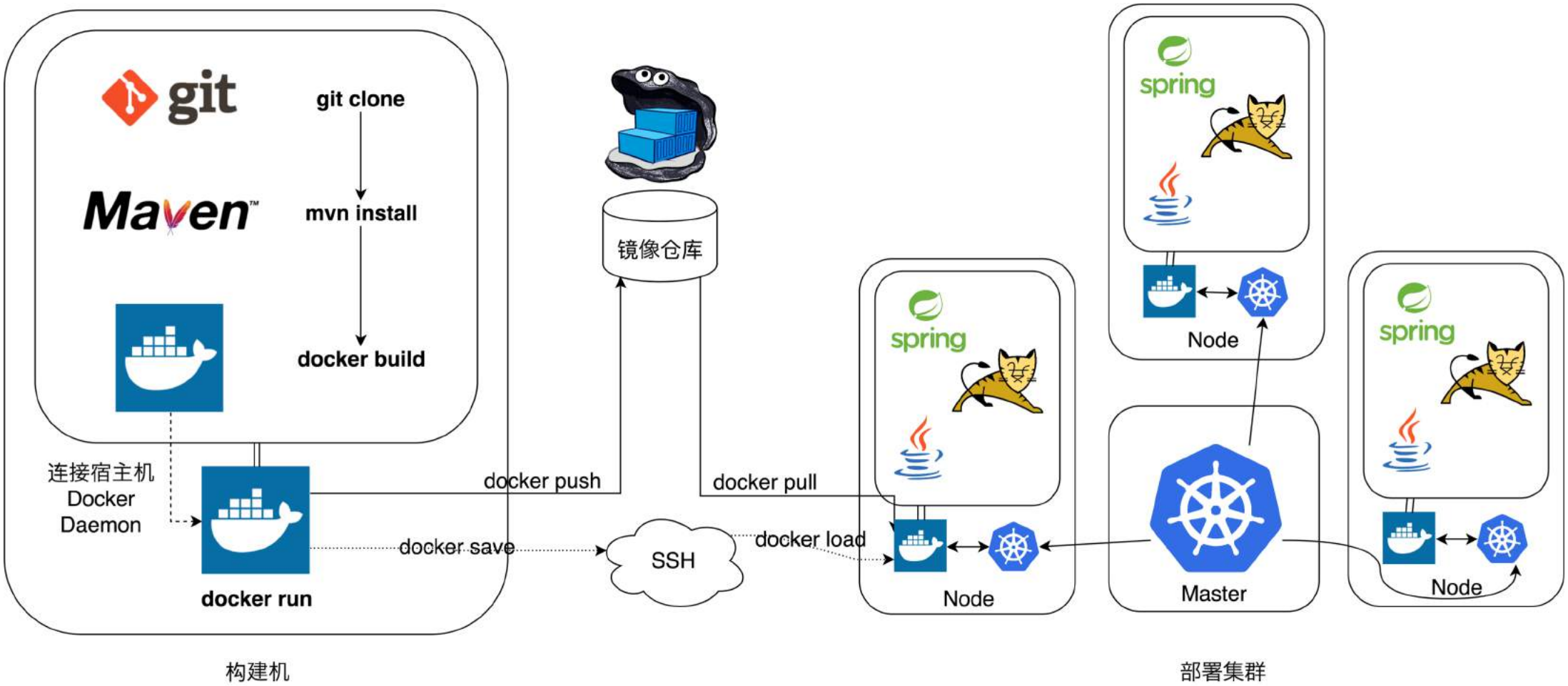
支持容器内多进程

- 通过 PID 1 对策进程防止 PID 1 相关特性导致多进程容器可能出现的僵尸进程问题

高可用性、容灾

- 镜像分发容灾
- Kubernetes 诸特性提升可用性

流程简图


git
git clone
Maven
mvn install
docker build
连接宿主机
Docker
Daemon
docker run
docker push
docker save
镜像仓库
SSH
docker pull
docker load
spring
Node
spring
Node
Master
spring
Node
构建机
部署集群

网易容器云平台容器化成果

60+服务

6个环境（包含三个线上region）

全部容器化部署

可用性提升

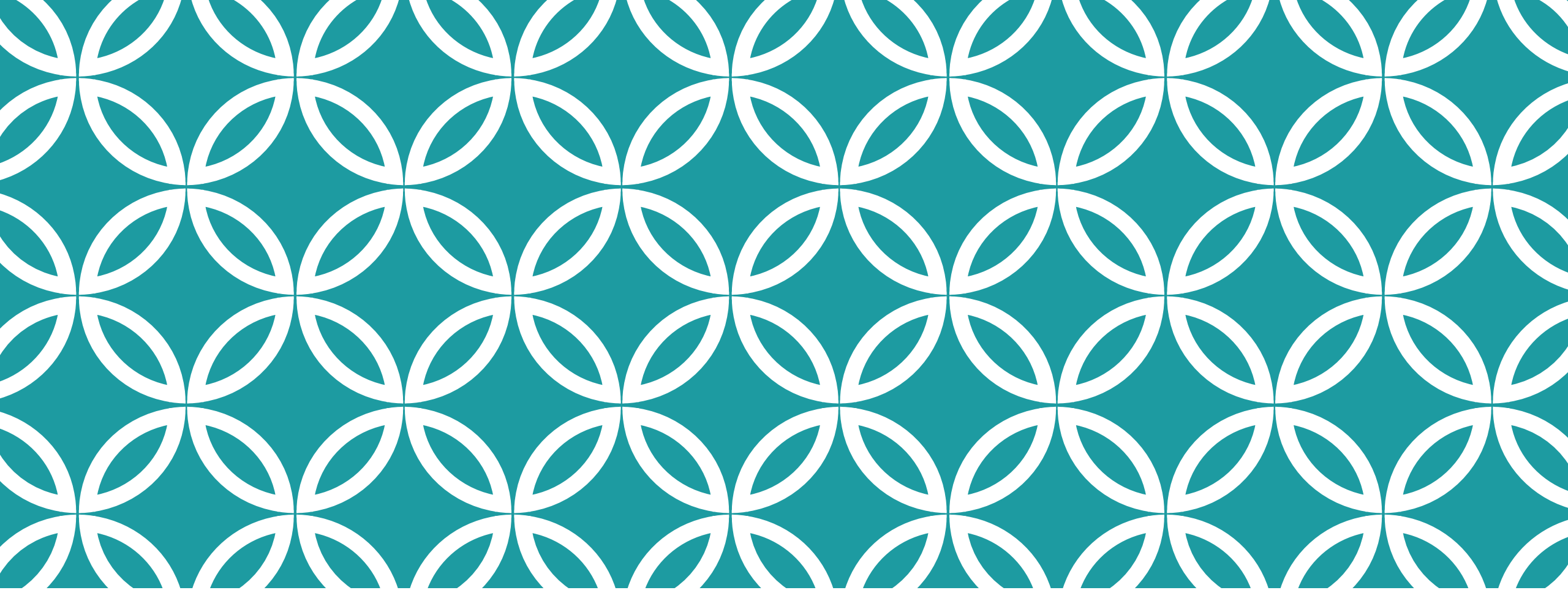
一键滚动升级

云计算基础服务
华东1
总览
容器服务
云服务器
负载均衡
镜像仓库
RDS
MongoDB
Redis
消息队列
云硬盘
对象存储
CDN

这里汇聚了你在网易云基础服务上的所有镜像仓库。 如何创建镜像仓库? 如何推送本地镜像?

+ 创建镜像仓库 | 全部 | 源码托管 | 镜像中心

仓库	更新时间	收藏	下载	版本	状态	设置
ncerepo/nce-web-callback	今天 15:17	0	254	118个版本	正常	设置
ncerepo/nce-interlayer-main	今天 15:06	0	508	212个版本	正常	设置
ncerepo/nce-container	今天 15:03	0	478	213个版本	正常	设置
ncerepo/nce-openapi	今天 11:22	0	822	339个版本	正常	设置
ncerepo/nce-api	今天 10:18	0	528	258个版本	正常	设置
ncerepo/nce-etcdwatcher	昨天	0	176	80个版本	正常	设置
ncerepo/nce-timer	2017-12-22	0	145	70个版本	正常	设置
ncerepo/bmm-web						



DOCKER基础知识和相关问题

DOCKER 基础知识

Docker 提供基于 namespace 和 cgroup 的容器化平台

- namespace 提供操作系统资源隔离（进程在宿主机上也是能看到的）
- cgroup 提供物理资源隔离

与 LXC 不同，Docker 面向应用，容器运行时并不会形成一个完整的系统，只会去运行 ENTRYPOINT + CMD 得到的命令

- ENTRYPOINT + CMD 会作为容器内的 PID 1 号进程

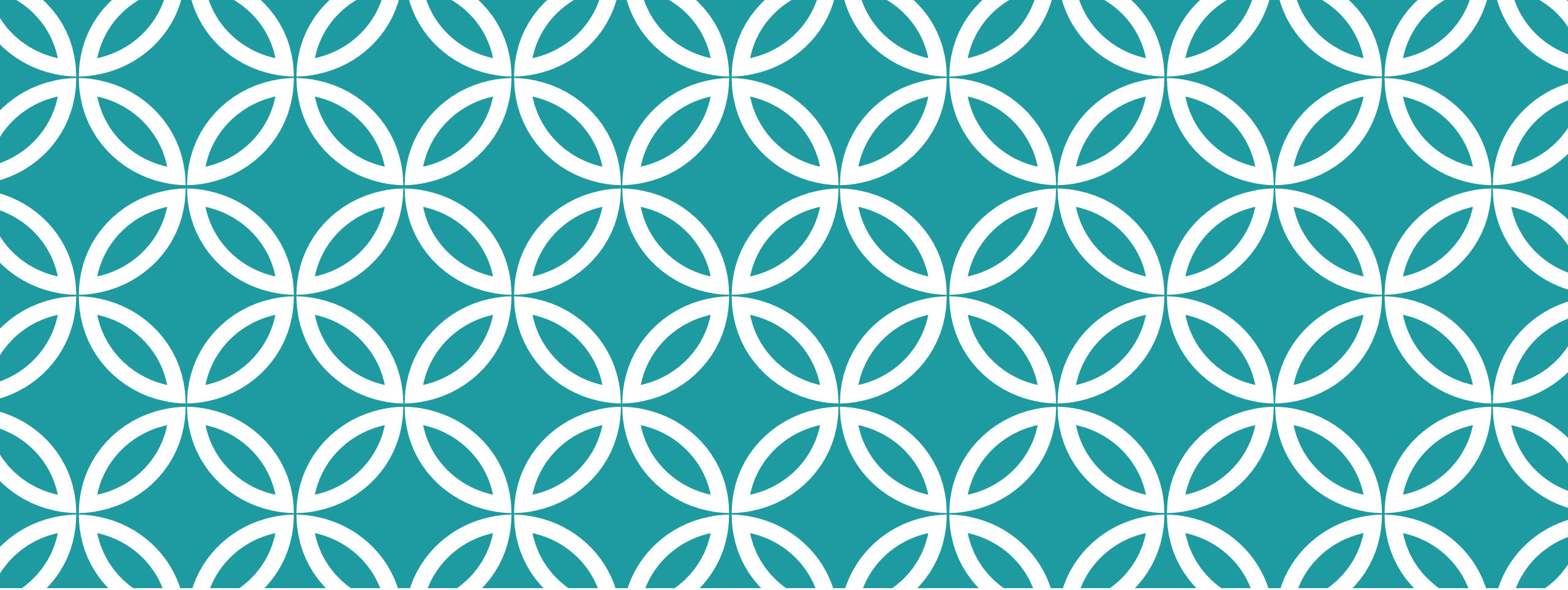
DOCKER 相关问题——PID 1

PID 为 1 的进程十分特殊

- 没有对如 SIGINT SIGTERM 等信号的默认行为
- 需要对 SIGCHLD 有正确的处理行为

如果容器需要跑多个进程，小心谨慎

- 可以采用 TINI
- 或者使用 dumb-init （容器服务目前使用，支持转发SIGINT等信号到所有进程）



容器镜像构建

容器镜像构建——镜像结构

Docker 镜像是分层的

- 如果两个镜像某些层一致，那么这些层只需要下载一份共享
- 如果基于旧有的镜像删除、更改一些文件，那么更改之前的内容由于位于之前已有的层，所以占用的空间不会减少

容器服务的实践

- 应用的镜像由两部分组成：基础镜像 + 应用

```
n:develop-20171228-20171225-145736
develop-20171228-20171225-145736: Pulling from ncerepo/nce-interlayer-main
1110dc31e100: Already exists
5ba3d77da3ce: Already exists
4825c429859e: Already exists
c71eb0978fb8: Already exists
37cb6fe623fc: Already exists
5329ff6837ed: Already exists
1e6e0af8e1cd: Already exists
15c4c798d388: Already exists
26412e346177: Pull complete
c2a6c12dd706: Pull complete
361359cdac33: Pull complete
Digest: sha256:5cf66198f0d72bce508a327d4df586c01ffe9bfe949973df6e60c58d18503185
Status: Downloaded newer image for hub.c.163.com/ncerepo/nce-interlayer-main:dev
```

容器服务应用镜像典型例子

容器镜像构建——多环境支持

不同环境下的配置是不同的，监听端口也可能是不同的（TOMCAT）

- 可考虑环境变量/启动参数处理

构建出来的镜像包含特制的启动用脚本处理上述问题

- 脚本可以通过读取环境变量获取相关信息
- 启动多进程（且包含管道等）使用脚本比较方便

容器镜像构建——构建用镜像

构建镜像当然可以通过 Dockerfile 直接构建

- 依赖管理问题

Docker 构建镜像依赖的是 Docker Daemon

- Docker 客户端直接通过 Unix Domain Socket 或 TCP 连接调用 Docker Daemon

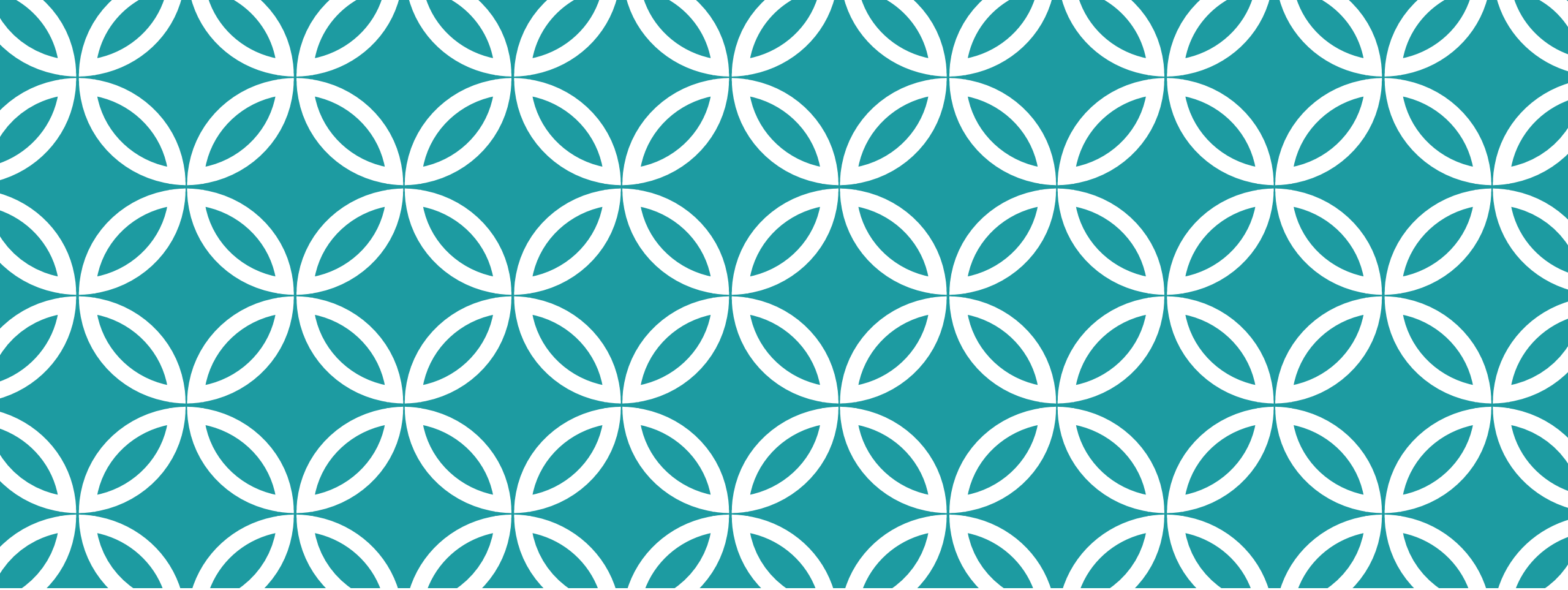
容器镜像构建——构建用镜像

构建用镜像包含代码管理、源码构建、镜像构建相关工具、依赖

- 代码管理：如 Git
- 源码构建：如 JDK Maven Cmake Go
- 镜像构建：如 Docker 独立客户端、相关 Dockerfile、组织用脚本等

构建发起方式例子

- 通过脚本封装 docker 调用，通过环境变量、运行参数等确定构建相关信息
- 利用 docker 官方客户端库编写相关应用



容器部署

容器部署——镜像分发

使用镜像仓库

- 直接使用 hub.c.163.com

备选方案——SSH (SCP)

- 在镜像仓库发生异常、与镜像仓库间网络发生异常时使用
- 只要能够保证从构建机到部署机存在可用网络通路且均支持 SSH，即可保证分发，提升可用性

容器部署——网络

Docker 提供了网络分离的手段，跨宿主机通过 NAT

- 由这篇文章看来，在延迟上不如 KVM vhost-net

Kubernetes 官方的网络方案中避免了容器间的 NAT，但一般会引入其他依赖

目前容器服务采用 host network 模式

- 将容器化的开销减小至内核级别的 namespace 隔离

容器部署进化

容器云平台的部署方式经历如下三个阶段：

- 实践初期，由于实验性、小规模使用，验证容器化管理服务的运行状况，在不引入过多依赖的前提下直接使用 Docker 命令完成小规模部署
- 验证完成后，为了方便在增加尽可能少的依赖的前提下逐步完成大规模部署，使用 Kubelet 加上 Static Pod 配置文件完成部署。
  - Static Pod 配置可采用 Git 管理
- 在对 Kubernetes 了解不断深入和线上使用经验不断丰富的前提下，为更全面的使用 Kubernetes 特性（如滚动升级、镜像密钥管理等），部署方式最终切换为 Kubernetes 集群

容器部署——DOCKER直接运行

Docker 本身也是支持在容器崩溃后自动重启的

- 适用于实验性/小规模使用

即便是在生产环境上使用，升级也可以通过脚本等方式简化

- 执行顺序：拉取、停旧容器、起新容器

大规模、多环境使用确实比较烦

容器部署——KUBELET

K8s 的 agent —— kubelet 也是可以独立运行的

- 使用 YAML/JSON 格式的 StaticPod 配置

对于上述的配置目录，可以通过 git 管理

无法使用除 Pod 以外的 K8s 资源类型

- 无法滚动更新

拉取镜像密钥需要在 /root/.docker/config.json 中存在

K

kubelet-conf

Star 2 Fork 0 SSH ssh://git@g.hz.netease.com:222 Global

Files (1.9 MB) Commits (503) Branches (10) Tags (0) Add Changelog Add License Add Contribution guide Set up CI

master kubelet-conf / History Find file

Update git2disconf.yaml
姜政冬 committed about 3 hours ago

1db9c1ef

Name	Last commit	Last Update
nbuild1.yq.pub.ntes	Update interlayer.yaml	a week ago
nbuild2.yq.pub.ntes	update imagecheck1	a week ago
nbuild3.yq.pub.ntes	update imagecheck2	a week ago
nce-manage1.yq.pub.ntes	update openapi m1	a week ago
nce-manage2.yq.pub.ntes	update openapi m2	a week ago

容器部署——KUBERNETES集群

通过搭建完整的 Kubernetes 集群处理

- Etcd 和 master 相关组件通过前述 StaticPod 启动
- 用 Deployment 启动 dashboard 方便运维查看

通过 namespace 区分不同环境

- Node 使用 label

使用 Deployment 部署应用

- 状态均保存于数据库，应用本身无状态

kubernetes

Search

+ CREATE

Workloads > Deployments

Cluster

- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace

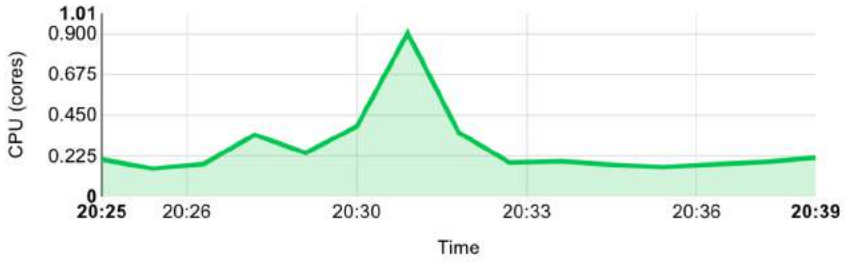
yq-liantiao

Overview

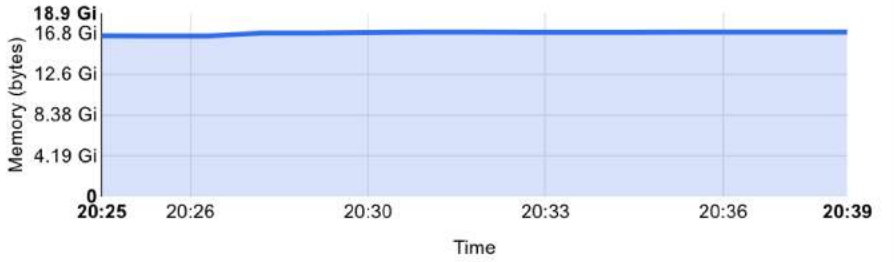
Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers

CPU usage



Memory usage



Deployments

Name	Labels	Pods	Age	Images
nce-gdashboard	nce-app: nce-gdashboard	1 / 1	a day	hub.c.163.com/ncerepo/nce-gdas
nce-res	nce-app: nce-res	1 / 1	a day	hub.c.163.com/ncerepo/nce-res:c
nce-interlayer-pubtest2	nce-app: nce-interlayer-pubtest2	1 / 1	11 days	hub.c.163.com/ncerepo/nce-inter
nce-interlayer-pubtest1	nce-app: nce-interlayer-pubtest1	1 / 1	11 days	hub.c.163.com/ncerepo/nce-inter
nce-interlayer	nce-app: nce-interlayer	1 / 1	15 days	hub.c.163.com/ncerepo/nce-inter

容器部署——滚动升级微调

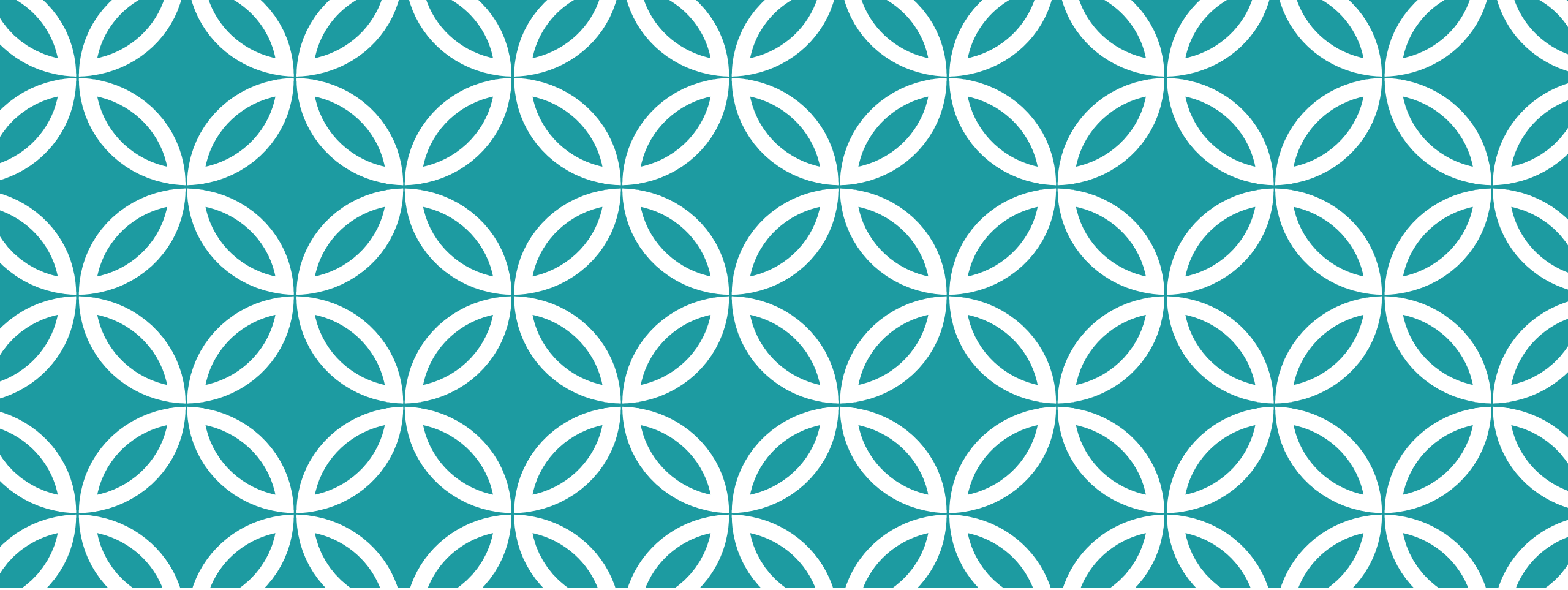
podSpec.container.lifecycle 设置让 K8s 在滚动升级时 Pod 启动前后自动做相应处理

- 分为 postStart 和 preStop

Deployment 的 minReadySeconds 设置

podSpec.container.readinessProbe 设置

Deployment 滚动更新在新启动的 Pod 状态处于 Ready 后才会进行下一步处理



CI/CD相关探索

CI/CD——容器镜像提测

容器镜像提测成为可能

传统提测流程，是针对代码的提测

- 开发、测试、上线都需要重新构建代码
- 无法保证每次得到的运行时二进制/字节码及依赖等状态都是一致的

容器镜像提测流程，是针对镜像的提测

- 只需构建一次
- 任何环境都能保证二进制/字节码、依赖等状态一致

减少环境不一致引发的问题

CI/CD自动化

目前还在探索中

自动化构建镜像既可以通过调用脚本、也可以通过试验性的 HTTP 服务完成

- 网络连通性问题

部署镜像可以通过调用 K8s 接口完成

谢谢！

Q&A

附：容器镜像构建具体流程

构建镜像中的构建流程

- 通过 git 拉取指定分支代码，支持可选的指定 commit hash
- 通过 Maven 构建，根据工程类型的不同（通过参数指定），采用不同的构建命令（子工程形式采用`mvn clean install -pl ${TARGET} -am`）
- 构建完毕后将 war 包取出、解压
- 通过 Dockerfile 将 war 包内容和相关配置、脚本构建为最终镜像
- 上传镜像（使用简易HTTP服务时不包含）

启动脚本流程

- 将 NCE_PORT 环境变量注入到 TOMCAT 的配置中，实现监听端口指定
- 使用 NCE_CONF 环境变量处理配置
  - 一般的为将 classpath 下名称和 NCE_CONF 一致的目录内容覆盖到 classpath 中
  - 对于镜像中依然保持 war 包形式的，在 TOMCAT 启动参数中注入
- 指定 XMX XMS 等，有默认值
- 将 NCE_JAVA_OPTS 环境变量注入 TOMCAT 启动参数中
- `catalina.sh run 2>&1 | tee /dev/stderr | rotatelog`