1、客户端高可用

微服务框架三个角色：Consumer（nacos-client），Provider（nacos-client），Registry（nacos-server）

nacos集群配置

a. 客户端重试

在请求之前会拿到nacos集群地址列表，在请求成功之前逐个重试，直到成功为止

b. 本地缓存文件Failover机制

1、nacos-client在接收到nacos-server的服务推动后，会在内存中保存一份，随后会落盘存储一份快照snapshot（这份快照可以保证本地的rpc调用即使在sever注册中心宕机后仍能够正常进行），这个本地缓存机制是默认关闭的，需要配置开启

缓存默认的存储路径：{USER_HOME}/nacos/naming/{namespace}

2、{USER_HOME}/nacos/naming/{namespace}下除了缓存文件之外还有一个failover文件夹，里面是跟snapshot一致的数据，用于可以人工修改以应对一些极端情况，猜测是snapshot内没有数据，客户端可以直接从这个文件夹读取到数据

c. 客户端naming通用参数

namingLoadCacheAtStart：启动时是否优先读取本地缓存，默认是关闭的

客户端缓存目录

客户端心跳的线程池大小，客户端定时轮询数据更新的线程池大小

是否打开https，默认是false

2、nacos两种健康检查模式

agent上报（client模式）

客户端通过心跳上报方式告知客户端（nacos注册中心）健康状态，默认心跳间隔5秒，

nacos会在超过15秒未收到心跳后将实例设置为不健康状态，超过30秒实例会被删除

这时对应的是临时实例，适用于弹性扩容的机器，流量降下来之后实例自己销毁就可以，不用手动去注销

服务端主动检测（客户端模式）

nacos主动探知客户端健康状态，默认间隔时间是20s，健康监测失败后会被标记为不健康，不会被删除

这时对应的是非临时实例的情况（需要提前手动注册）

临时实例

- 临时实例不会再nacos客户端持久化存储，需要通过上报心跳的方式包活
- 如果一段时间没有上报心跳，则会重新将这个实例注册
- 对应的持久化实例则会被nacos客户端存储，即使客户端进行不存在了，这个实例也不会被删除，只会被标记为不健康
- 同一个服务下可以同时存在临时实例和持久化实例，这意味着即使所有的实例进程不在时，会有部分从服务摘除，说呢过下的则会保留

- 是否临时实例使用参数ephemeral来判断，true对应的是client模式，false是server模式
- Nacos 1.0.0之前用服务端的健康检查模式来区分这几种（client、server 和none，客户端上报，服务探测，取消健康检查），之后则使用实例的`ephemeral参数来判断`
- 临时服务使用的是 Nacos 为服务注册发现场景定制化的私有协议 distro，其一致性模型是 AP

```
#false为永久实例，true表示临时实例开启，注册为临时实例
spring.cloud.nacos.discovery.ephemeral=true
```

3、nacos server运行模式

CAP定理

分布式系统中，Consistency（一致性）,Avaliability（可用性）,Partittion tolerance（分区容错性） 三者不可兼得，要么AC，要么CP，要么AP

- 一致性（C）：分布式系统中所有数据备份，在同一时刻是否相同（所有节点访问同一份数据的最新数据副本）
- 可用性（A)：集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求，即数据高可用
- 分区容忍性（P)：分区相对于通信的时限要求，系统如不能在时限内保证数据一致性，就意味着发生了分区的情况，就得在C和A治之间做出选择
- 分区：分布式系统分为很多部分，当出现一些部分之间不能通信，就说出现了网络分区

AP模式

为了可用性降低了一致性（不支持数据一致性），因此AP模式支持注册临时实例，在网络分区下也能注册实例，在AP，偶是下不能编辑服务的元数据，但是允许创建一个默认配置的服务，注册实例前不需要创建服务的操作，这个模式下其实是降级成了一简单的字符串标识，不存在任何属性，会在注册实例的时候自动创建

AP模式下的distro协议

1. nacos启动时首先从其他远程节点同步全部数据
2. 每个节点是平等的都可以处理写入请求，同时把新数据同步到其他节点
3. 每个节点只负责部分数据，定时发送自己负责数据的校验值到其他节点来保持数据一致性

节点收到写请求

1、当该节点接收到的属于该节点负责的服务时，直接写入，并定时同步到其他节点

2、当节点收到不属于该节点负责的服务时，将在集群内部路由，转发给负责节点，进行写入

3、节点宕机后，该节点负责的写入任务会被转移到其他节点

节点收到读请求

读请求不需要路由，集群中每个几点都会同步服务状态，每个节点都有最新的数据

CP模式

Cp模式支持注册持久实例，此时以Raft协议为集群运行模式，因此网络分区下不能注册实例，可以编辑服务器的配置，注册实例前必须先注册服务，如果服务不存在则会返回错误

CP模式下会有raft协议

MIXED模式

能够同时支持临时实例和持久化实例的注册，注册实例前必须先创建服务，服务已经存在的前提下，临时实例可以在网络分区的情况下注册

4、集群内部心跳同步结构

nacos中，出于可用性的考虑，一个心跳报文包含了全部的服务信息，这样相比仅仅发送探测信息降低了吞吐量，但是提高了可用性

两种场景

i. server节点全部宕机，服务数据全部丢失，即使server恢复也丢失了数据，而心跳包含全部内容可以在心跳期间就恢复出服务，保证可用性

ii. server出现网络分区，由于心跳可以创建服务，在极端网络故障下也可以保证基础的可用性

5、集群部署模式高可用

节点数量（待深入）

```
Nacos 有两个一致性协议：distro 和 raft，distro 协议不会有脑裂问题，所以理论来说，节点数大于等于 2 即可；raft 协议的投票选举机制则建议是 2n+1 个节点。
综合来看，选择 3 个节点是起码的，其次处于吞吐量和更吞吐量的考量，可以选择 5 个，7 个，甚至 9 个节点的集群
```

多可用区部署

各节点之间网络延时不能很高，否则会影响数据同步

各节点所处机房，可用区应尽量分散，以避免单点故障

部署模式

建议K8s部署，或者阿里云的ecs部署

高可用的nacos部署架构

高可用nacos的部署实操

参考连接：https://www.cnblogs.com/crazymakercircle/p/15393171.html#autoid-h3-7-0-2