

Docker Distribution二次开发实践

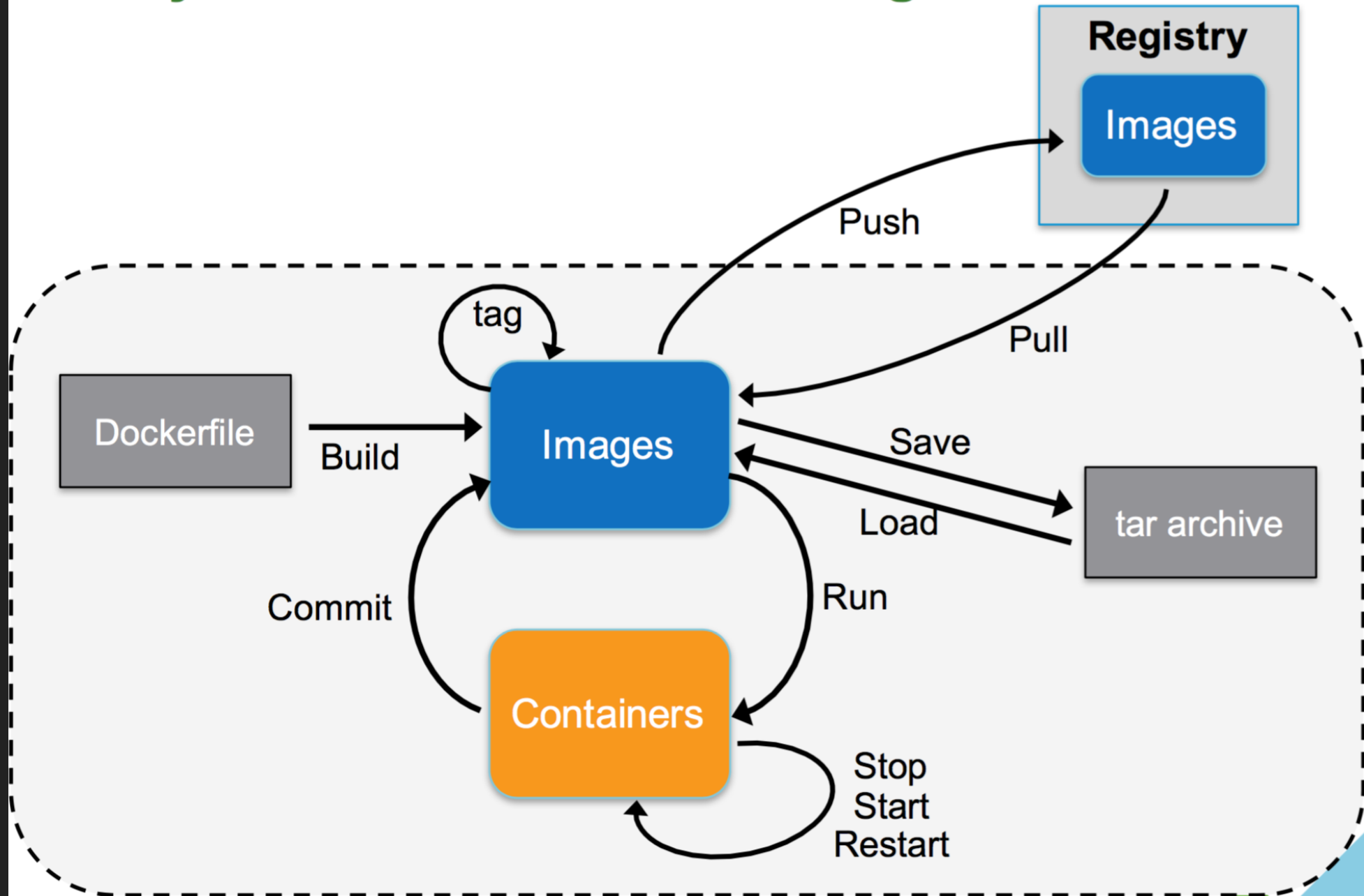
1. 基本概念介绍、理解
 2. 源码架构剖析
 3. 支持S3多bucket存储驱动实现
 4. 测试性能对比
-
5. 总结问题



基本概念

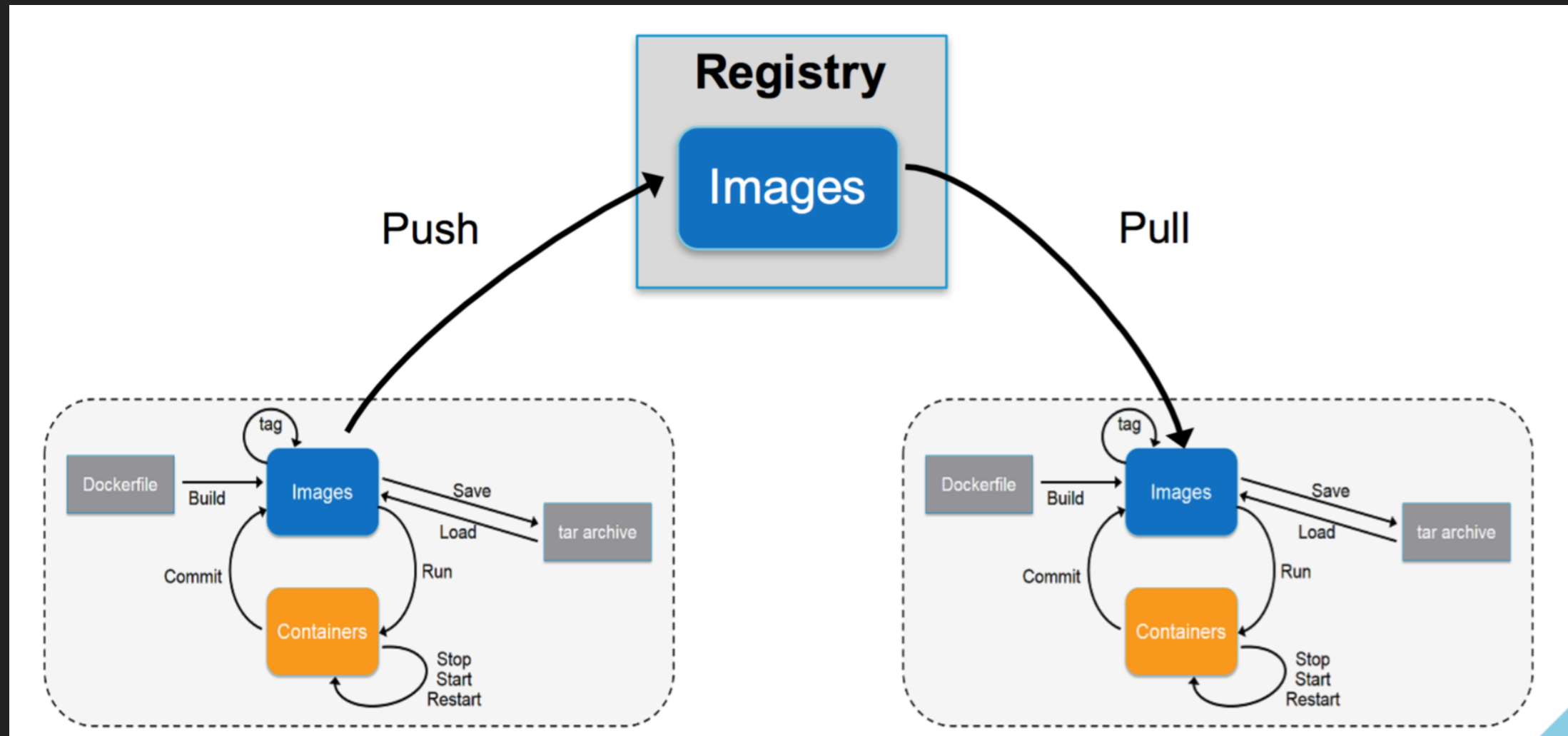
Build - Ship - Run

Lifecycle of Containers and Images

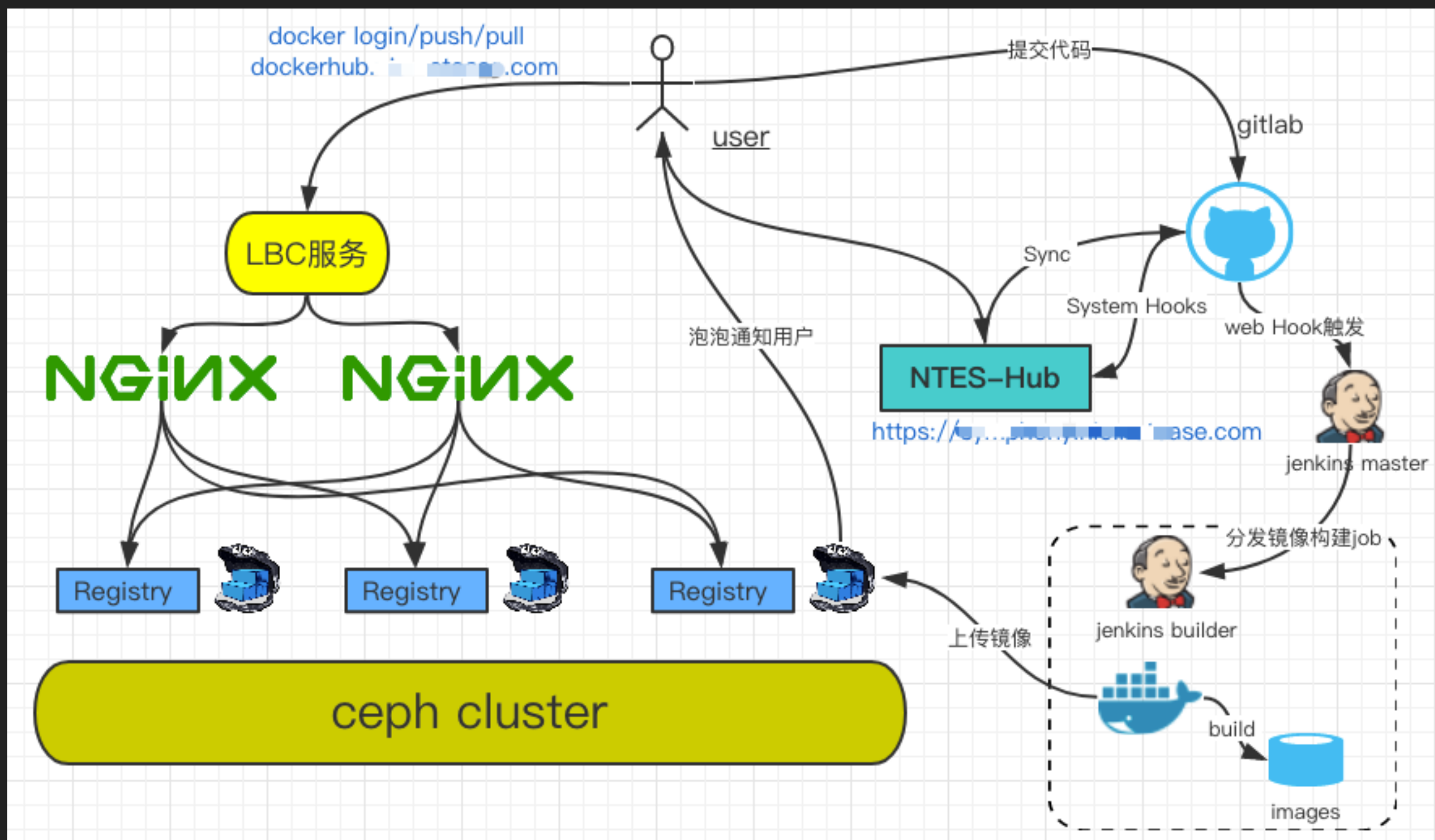


基本概念

Registry - 管理镜像的核心组件

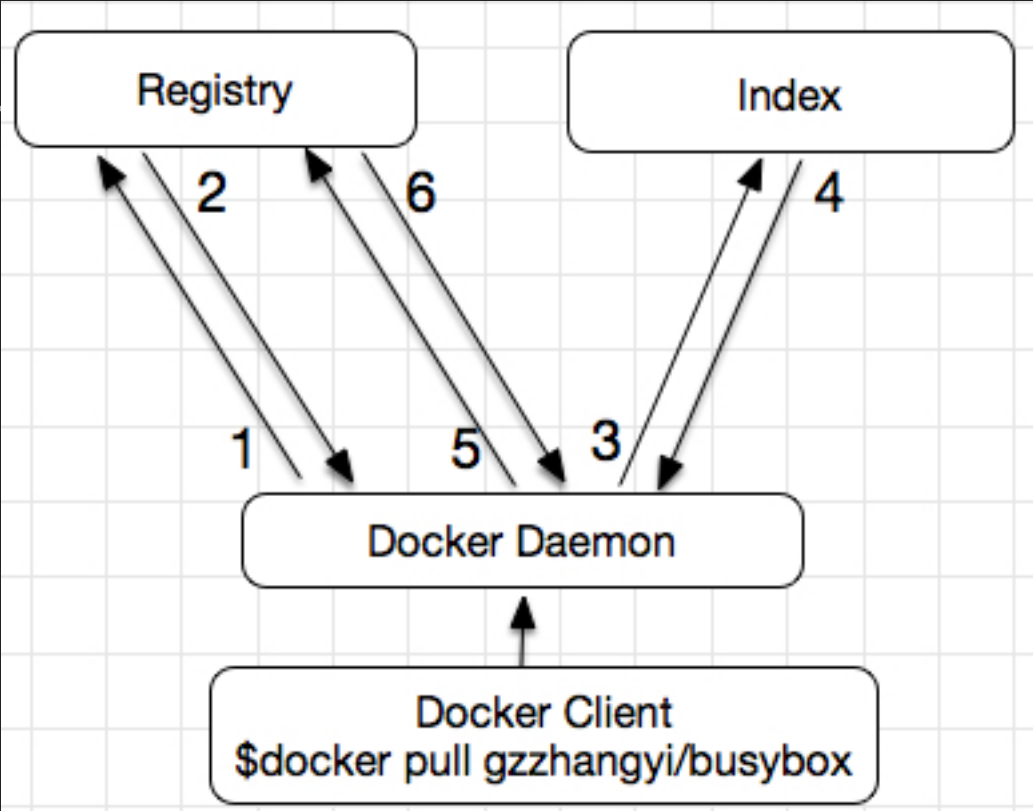


高可用镜像仓库服务架构



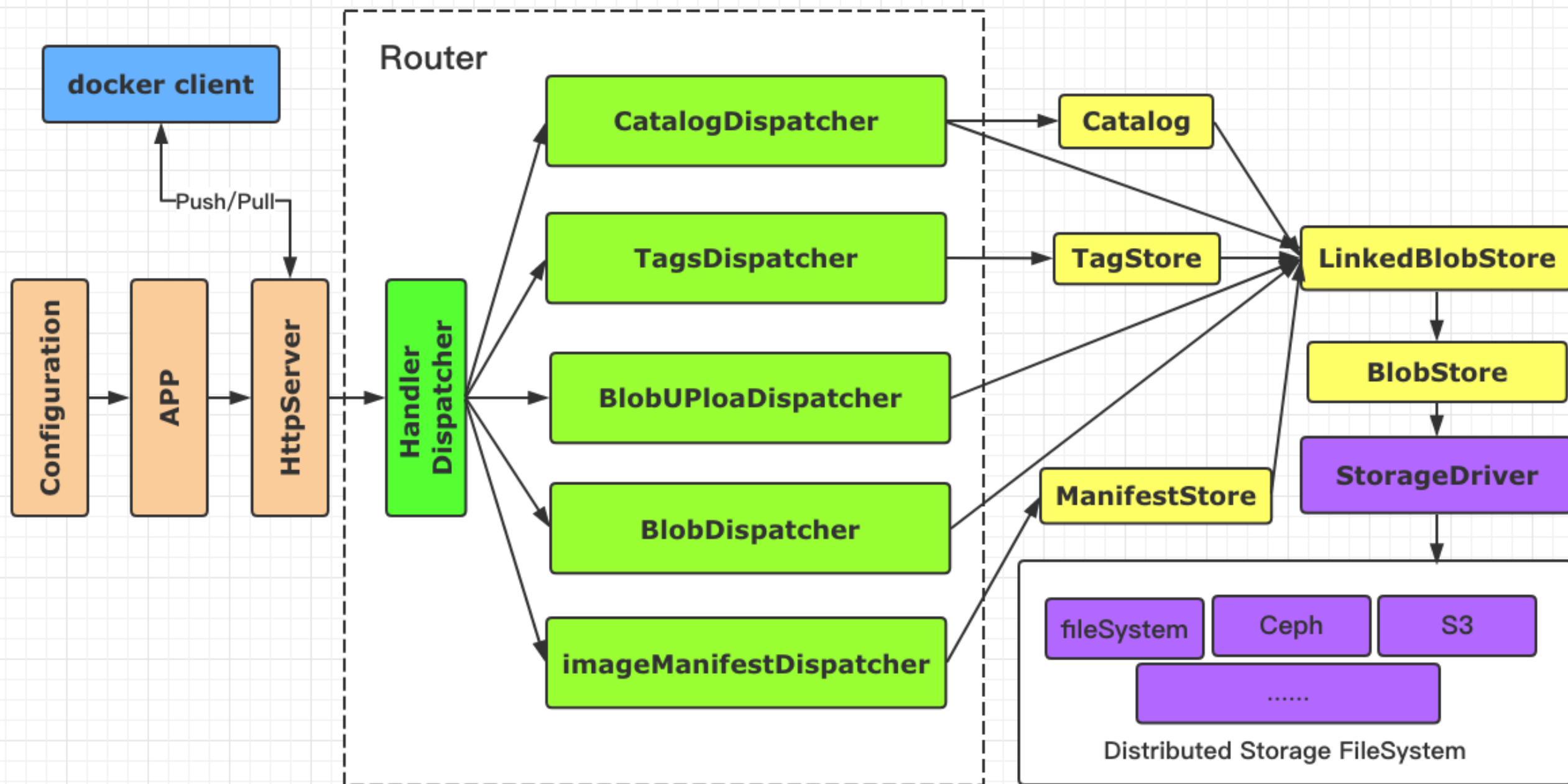
镜像仓库管理页面

- 基于RBAC的权限用户管理
- 公有\私有镜像权限区分
- 镜像信息展示



7	0.000...	10.160.248.11	10.160.248.12	HTTP	66	HTTP/1.1	400	Bad Request	(text/plain)
19	0.001...	10.160.248.11	10.160.248.12	HTTP	66	HTTP/1.1	400	Bad Request	(text/plain)
28	0.001...	10.160.248.12	10.160.248.11	HTTP	240	GET	/v2/	HTTP/1.1	
30	0.004...	10.160.248.11	10.160.248.12	HTTP	430	HTTP/1.1	401	Unauthorized	(application/json)
41	0.005...	10.160.248.11	10.160.248.12	HTTP	66	HTTP/1.1	400	Bad Request	(text/plain)
53	0.006...	10.160.248.11	10.160.248.12	HTTP	66	HTTP/1.1	400	Bad Request	(text/plain)
62	0.006...	10.160.248.12	10.160.248.11	HTTP	240	GET	/v2/	HTTP/1.1	
64	0.009...	10.160.248.11	10.160.248.12	HTTP	430	HTTP/1.1	401	Unauthorized	(application/json)
72	0.050...	10.160.248.12	10.160.248.11	HTTP	1136	HEAD	/v2/zhangyi/debian/blobs/sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d		
74	0.236...	10.160.248.11	10.160.248.12	HTTP	244	HTTP/1.1	404	Not Found	
82	0.253...	10.160.248.12	10.160.248.11	HTTP	1115	POST	/v2/zhangyi/debian/blobs/uploads/	HTTP/1.1	[Packet size limited during capture]
84	0.633...	10.160.248.11	10.160.248.12	HTTP	633	HTTP/1.1	202	Accepted	
92	0.633...	10.160.248.12	10.160.248.11	HTTP	1510	PUT	/v2/zhangyi/debian/blobs/uploads/86c02bbe-9a33-4769-83f5-a0ca2506965b?_state=xsroCxEqUp5L5w5jWBg		
94	1.487...	10.160.248.11	10.160.248.12	HTTP	463	HTTP/1.1	201	Created	
102	1.488...	10.160.248.12	10.160.248.11	HTTP	1136	HEAD	/v2/zhangyi/debian/blobs/sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d		
104	1.988...	10.160.248.11	10.160.248.12	HTTP	461	HTTP/1.1	200	OK	
112	1.988...	10.160.248.12	10.160.248.11	HTTP	1136	HEAD	/v2/zhangyi/debian/blobs/sha256:4d690fa986553fb89f8ea3131e923ed9470d7a863add7991ea547805d5cab0d		
114	1.995...	10.160.248.11	10.160.248.12	HTTP	244	HTTP/1.1	404	Not Found	
122	14.72...	10.160.248.12	10.160.248.11	HTTP	1115	POST	/v2/zhangyi/debian/blobs/uploads/	HTTP/1.1	[Packet size limited during capture]
124	14.94...	10.160.248.11	10.160.248.12	HTTP	631	HTTP/1.1	202	Accepted	
132	14.94...	10.160.248.12	10.160.248.11	HTTP	4162	PUT	/v2/zhangyi/debian/blobs/uploads/e0e6238d-490a-40a6-95e8-4bc18bd078c7?_state=dpCE70itbD1epINaKK		
502	15.07...	10.160.248.12	10.160.248.11	HTTP	17442	Continuation	[Packet size limited during capture]		
827	15.10...	10.160.248.12	10.160.248.11	HTTP	17442	Continuation	[Packet size limited during capture]		
1050	15.11...	10.160.248.12	10.160.248.11	HTTP	17442	Continuation	[Packet size limited during capture]		
1524	15.15...	10.160.248.12	10.160.248.11	HTTP	8754	Continuation	[Packet size limited during capture]		
2201	15.21...	10.160.248.12	10.160.248.11	HTTP	17442	Continuation	[Packet size limited during capture]		
2484	15.23...	10.160.248.12	10.160.248.11	HTTP	3746	Continuation	[Packet size limited during capture]		
4224	15.76...	10.160.248.12	10.160.248.11	HTTP	13098	Continuation	[Packet size limited during capture]		
4934	15.81...	10.160.248.12	10.160.248.11	HTTP	15994	Continuation	[Packet size limited during capture]		
5038	16.81...	10.160.248.11	10.160.248.12	HTTP	463	HTTP/1.1	201	Created	
5046	16.83...	10.160.248.12	10.160.248.11	HTTP	4162	PUT	/v2/zhangyi/debian/manifests/7.9	HTTP/1.1	[Packet size limited during capture]
5050	17.46...	10.160.248.11	10.160.248.12	HTTP	467	HTTP/1.1	201	Created	

Docker Distribution 源码架构



源码解析三部曲 – 理解第三方库

`github.com/spf13/cobra` :

`github.com/docker/libtrust` :

`golang.org/x/net/context` :

`github.com/Sirupsen/logrus` :

`net/http` :

`github.com/gorilla/handlers` :

源码解析三部曲 – 理解抽象

ManifestService 描述提供镜像manifests的操作

type ManifestService interface {

Exists(ctx context.Context, dgst digest.Digest) (bool, error)

Get(ctx context.Context, dgst digest.Digest, options ...ManifestServiceOption) (Manifest, error)

Put(ctx context.Context, manifest Manifest, options ...ManifestServiceOption) (digest.Digest, error)

Delete(ctx context.Context, dgst digest.Digest) error

}

TagService 提供访问已标记对象的信息的接口

type TagService interface {

Get(ctx context.Context, tag string) (Descriptor, error)

Tag(ctx context.Context, tag string, desc Descriptor) error

Untag(ctx context.Context, tag string) error

All(ctx context.Context) ([]string, error)

Lookup(ctx context.Context, digest Descriptor) ([]string, error)

}

源码解析三部曲 – 理解抽象

```
## BlobService 组合 对远程blob服务的读、写操作 的抽象
type BlobService interface {
    BlobStatter  让blob descriptor可通过digest描述
    BlobProvider 描述获取blob数据的操作
    BlobIngestor 获取blob数据
}

## BlobStore 组合了 完整地 与blob相关操作的抽象，实现了包括对blob的
type BlobStore interface {
    BlobService
    BlobServer 通过http serve blob
    BlobDeleter 使能blob删除接口
}

# Descriptor 结构用来fetch、store、target任何blob,该结构也用来描述协议格式
type Descriptor struct {

    MediaType string `json:"mediaType,omitempty"`

    Size int64 `json:"size,omitempty"`

    Digest digest.Digest `json:"digest,omitempty"`

    URLs []string `json:"urls,omitempty"`

}
```

源码解析三部曲 – 理解流程

前提

- ▶ 镜像在distribution的存储目录说明

- ▶ blobs目录：

- ▶ 存储每层数据

- ▶ repositories目录：

- ▶ 存放镜像仓库中的镜像的组织信息

```
.
├── blobs
│   └── sha256
│       ├── 05
│       │   └── 052b6b7ef61d6cb585d0be739a46c3bbd1e4d8dfbe47e68fff2b97e9177de7a6
│       │       └── data
│       ├── 59
│       │   └── 5990b2a5d0cf3fdf7f359858ce7af2e4c89dcdbbf77aec53f58d7fac0578b01d
│       │       └── data
│       ├── 67
│       │   └── 674ded4e0a754b70be8f9eabf401db21d6caaa2aba6305bcebdffb67ea7e0424
│       │       └── data
│       ├── 7b
│       │   └── 7bccea55ff9419d79f0ba121bd86e046f83cffa253444ae7bf7731e58cccf4d3
│       │       └── data
│       ├── 7f
│       │   └── 7fc13f83063b62bed7375560fb34b9ccd8f8bcb91b40c70080dcc95bb1e7dc1b
│       │       └── data
│       ├── a3
│       │   └── a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
│       │       └── data
│       ├── ad
│       │   └── adc97f4d0e44492ecf37b03804353142c0109eadd6dcbae9a47c0b837eeebded
│       │       └── data
│       ├── c4
│       │   └── c4ba5c6a7c94eca6f6c91c7c76659f054b58d7398197b48821df0e34b07d4157
│       │       └── data
│       ├── d6
│       │   └── d6af8f7e802dcc5b9cc3e1b293ece0dbd7ec73506fd160adb4d2cc9cee9f9884
│       │       └── data
│       └── fe
│           └── fe76480c154386f50ee75a2a4869992b320db813d1d62c3d5975a38079fd6654
│               └── data
```

源码解析三部曲 – 理解流程

前提

镜像在distribution的存储目录说明

blobs目录：

存储每层数据

repositories目录：

存放镜像仓库中的镜像的组织信息



源码解析三部曲 – 理解流程

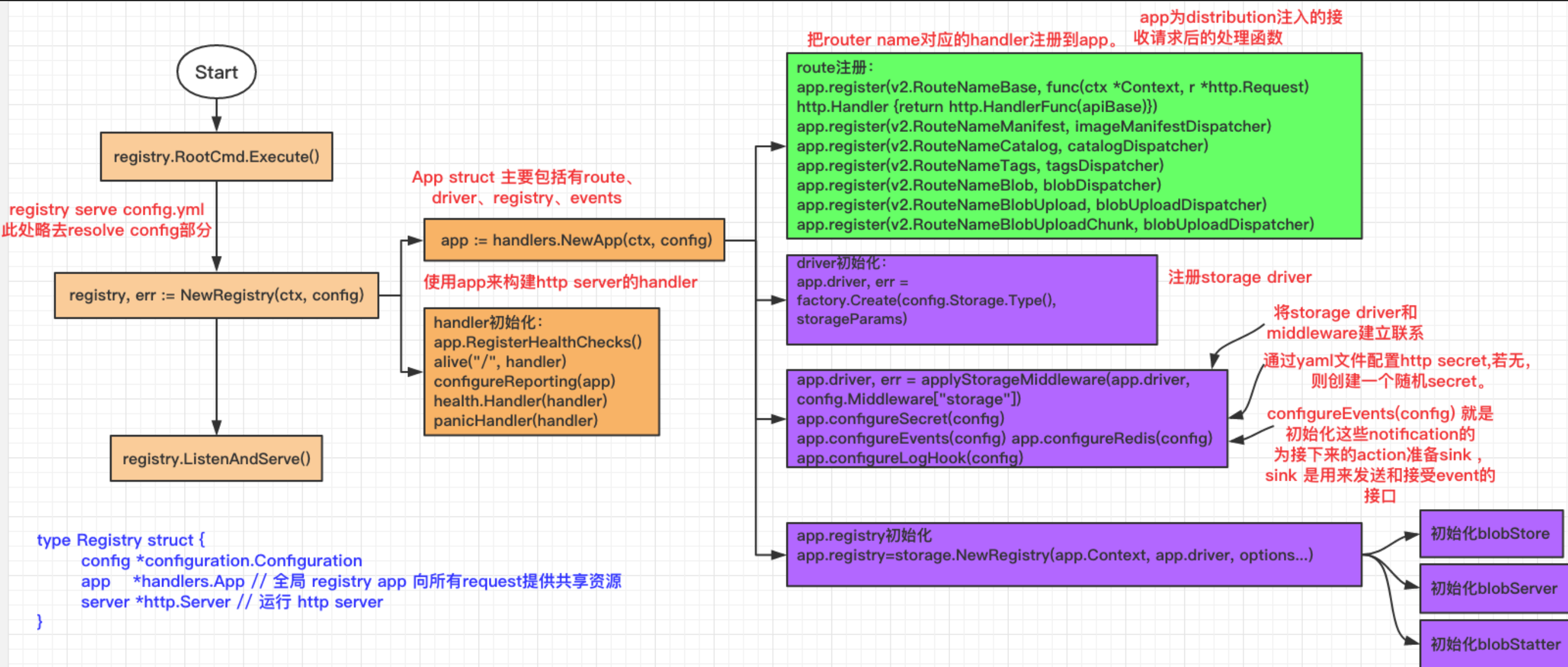
▶ 启动和初始化

```
type App struct {
    context.Context

    Config *configuration.Configuration
    //main application router, configured with dispatchers
    router      *mux.Router
    //driver maintains the app global storage driver instance.
    driver      storagedriver.StorageDriver
    //registry is the primary registry backend for the app instance.
    registry    distribution.Namespace
    accessController auth.AccessController // main access controller for application
    httpHost url.URL
    events struct { // registry事件的相关配置
        sink    notifications.Sink
        source notifications.SourceRecord
    }
    redis *redis.Pool
    trustKey libtrust.PrivateKey
    isCache bool
    readOnly bool
}
```

源码解析三部曲 – 理解流程

启动和初始化



支持多bucket S3存储驱动实现

▶ Storagedriver.go中接口函数

▶ Name/GetContent/PutContent

▶ Reader/Writer

▶ Storagedriver / factory包

▶ factory.register

▶ factory.create

```
type StorageDriver interface {
    Name() string
    // 通过存储path获取内容, 并存储在[]byte, 主要用来获取小的objects.
    GetContent(ctx context.Context, path string) ([]byte, error)

    // 通过path路径来上传内容到[]byte, 主要用来存放小的objects.
    PutContent(ctx context.Context, path string, content []byte) error

    // 通过给定的offset来获取存放在path中的内容
    Reader(ctx context.Context, path string, offset int64) (io.ReadCloser, error)

    // 根据path存放的内容, 把数据提交到 FileWriter返回
    Writer(ctx context.Context, path string, append bool) (FileWriter, error)

    // 通过给定的path来获取FileInfo
    Stat(ctx context.Context, path string) (FileInfo, error)

    // 根据给定的path来返回一系列的objects
    List(ctx context.Context, path string) ([]string, error)

    Move(ctx context.Context, sourcePath string, destPath string) error

    Delete(ctx context.Context, path string) error

    URLFor(ctx context.Context, path string, options map[string]interface{}) (string, error)
}
```

支持多bucket S3存储驱动实现

```
// FileWriter 提供了在后端存储的类似文件对象的抽象、FileWriter会把所有写的内容flush到Close,
type FileWriter interface {
    io.WriteCloser

    // Size returns the number of bytes written to this FileWriter.
    Size() int64

    // Cancel removes any written content from this FileWriter.
    Cancel() error

    // Commit flushes all content written to this FileWriter and makes it
    // available for future calls to StorageDriver.GetContent and
    // StorageDriver.Reader.
    Commit() error
}
```

▶ S3存储模型

- ▶ Bucket: 用于存储对象的容器，名称全局唯一。
- ▶ Object: 所有资源的最终存放方式。
 - ▶ 元数据对应关系： key name → 文件名称 , etag → 文件md5

支持多bucket S3存储驱动实现

- ▶ 单个bucket大小 $\leq 2T$
- ▶ 单个bucket拥有object数量 不建议超过100w个

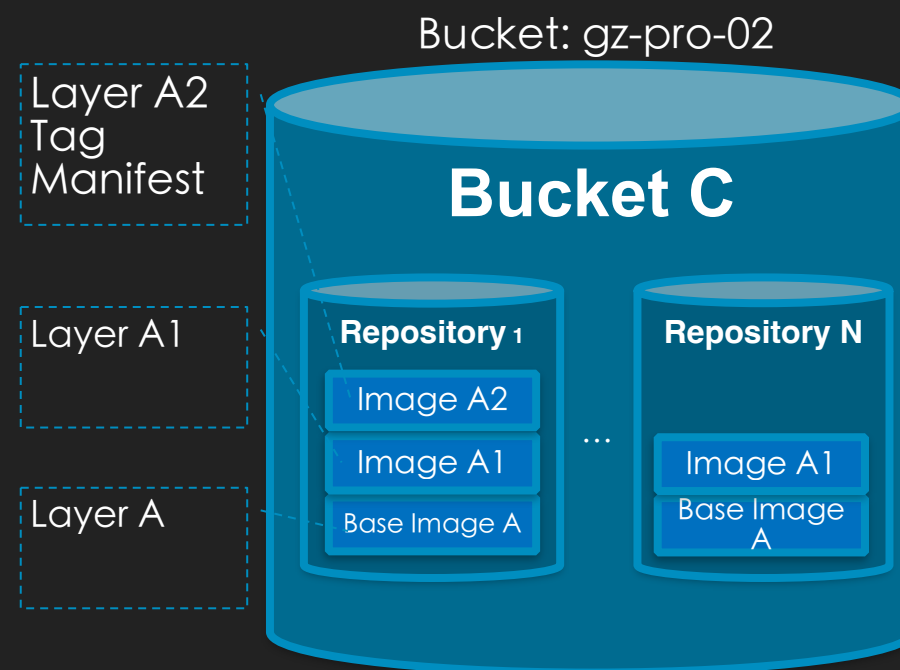
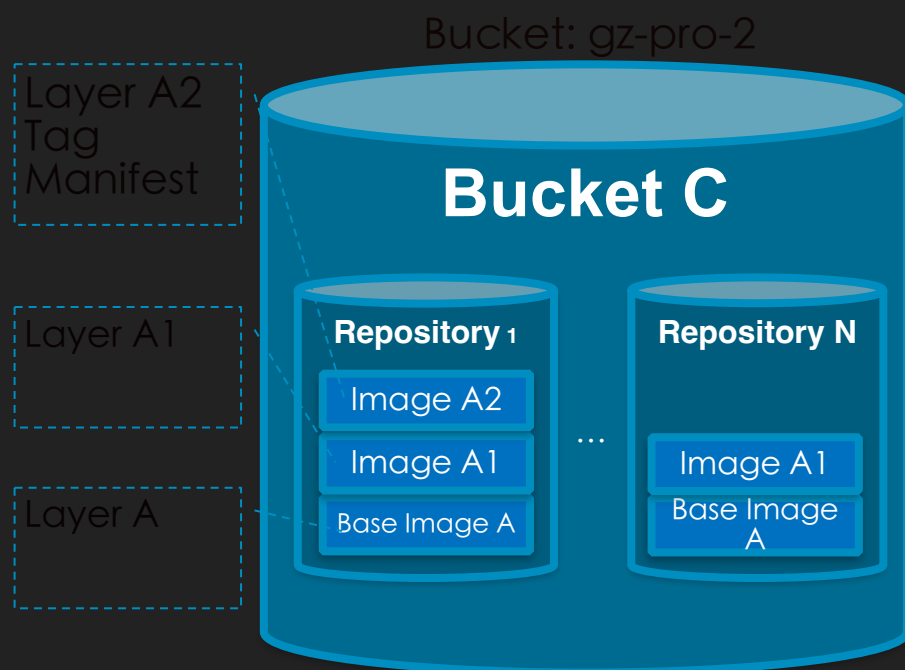
```
[<Bucket: hdg4-test01>, <Bucket: hdg4-test02>]
```

```
<Bucket: hdg4-test01>
```

```
[<Key: <Bucket: hdg4-test01>,docker/registry/v2/blobs/sha256/4e/4efd5033df5ddf3cff692505ae2a8e6d492014dbadd2f7281ccda9f744d41901/data>,  
<Key: <Bucket: hdg4-test01>,docker/registry/v2/repositories/library/busybox/_manifests/revisions/sha256/4efd5033df5ddf3cff692505ae2a8e6d492014dbadd2f7281ccda9f744d41901/link>,<Key: <Bucket: hdg4-test01>,docker/registry/v2/repositories/library/busybox/_manifests/tags/latest/index/sha256/4efd5033df5ddf3cff692505ae2a8e6d492014dbadd2f7281ccda9f744d41901/link>]
```

```
<Bucket: hdg4-test02>
```

```
[<Key: <Bucket: hdg4-test02>,docker/registry/v2/blobs/sha256/46/465de20efd1176e51a2b66f135efd06a46ac82b2ac2249d05222afaeb39dcf40/data>,  
<Key: <Bucket: hdg4-test02>,docker/registry/v2/blobs/sha256/f9/f9b6f7f7b9d34113f66e16a9da3e921a580937aec98da344b852ca540aaa2242/data>,<Key: <Bucket: hdg4-test02>,docker/registry/v2/repositories/library/busybox/_layers/sha256/465de20efd1176e51a2b66f135efd06a46ac82b2ac2249d05222afaeb39dcf40/link>,<Key: <Bucket: hdg4-test02>,docker/registry/v2/repositories/library/busybox/_layers/sha256/f9b6f7f7b9d34113f66e16a9da3e921a580937aec98da344b852ca540aaa2242/link>,<Key: <Bucket: hdg4-test02>,docker/registry/v2/repositories/library/busybox/_manifests/tags/latest/current/link>]
```



支持多bucket S3存储驱动实现

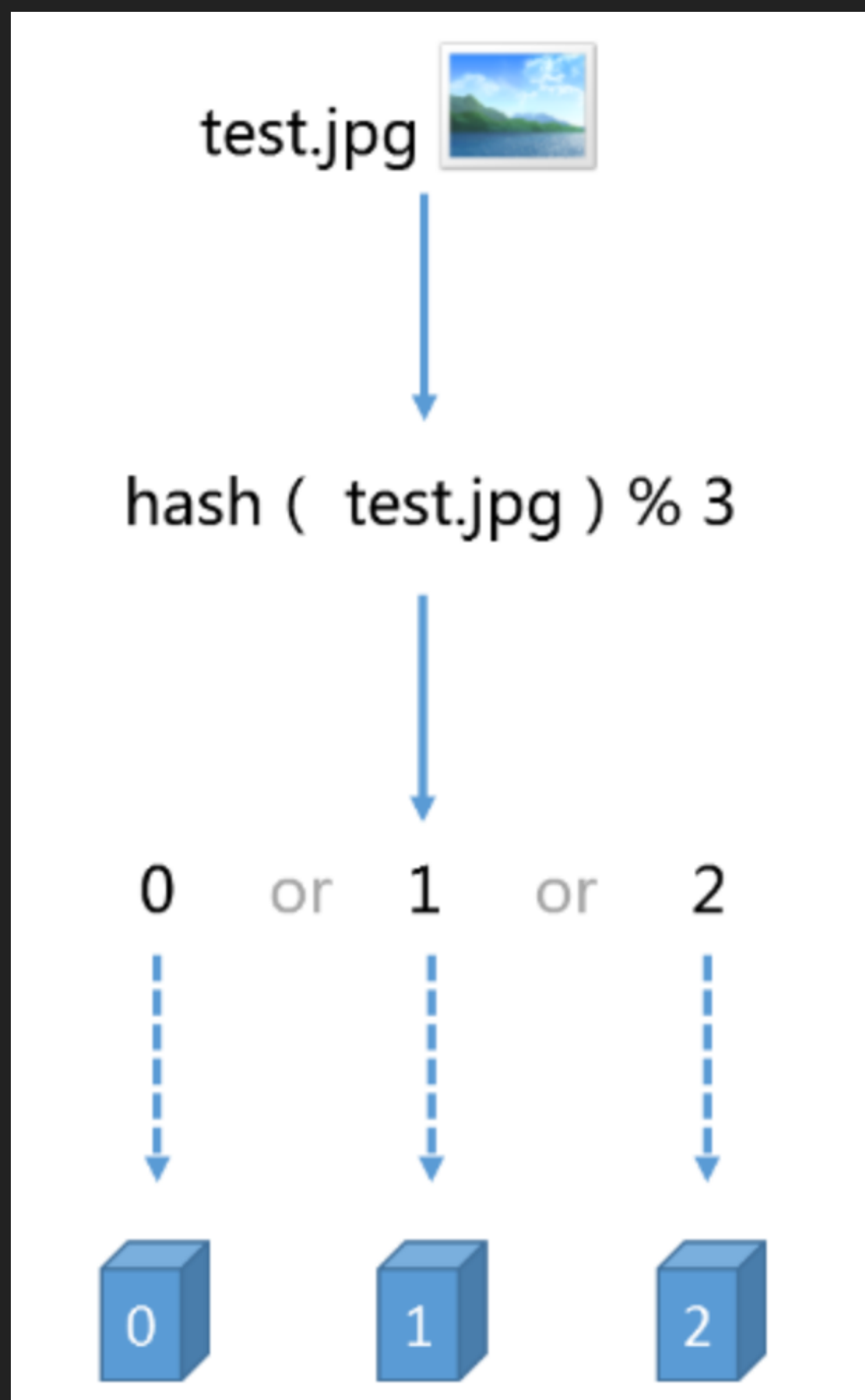
理解场景

▶ 普通存kv数据库的方式

- ▶ 1. 增：当查询到bucket满，增加bucket. 每次存数据都要查kv DB.
- ▶ 2. 删：当删除数据时，删映射(开销)，旧bucket空间浪费。
- ▶ 3. bucket宕掉：分布式场景。
 - ▶ 影响全局。其他镜像的某一层可能在此bucket中。
 - ▶ 故障恢复。恢复数据多且慢。
- ▶ 4. kv DB本身的高可用十分关键。且操作频繁，开销需要考虑。

支持多bucket S3存储驱动实现

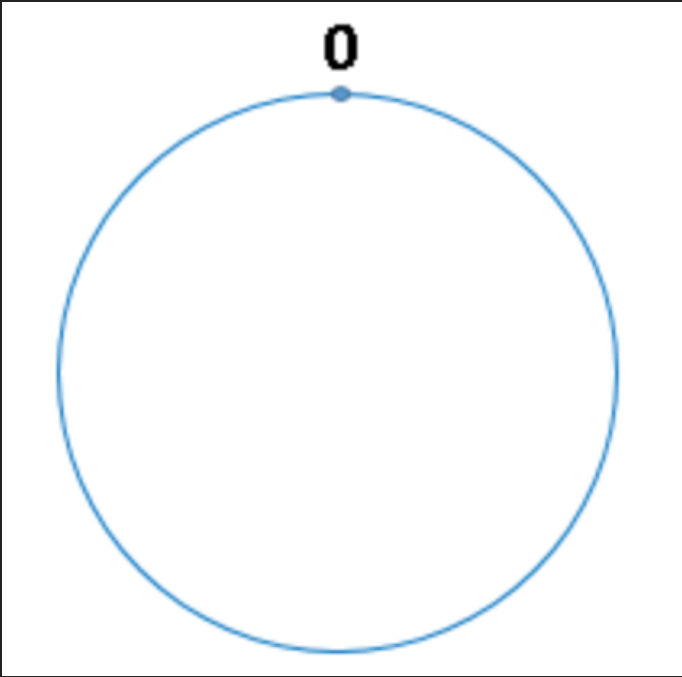
普通的哈希算法



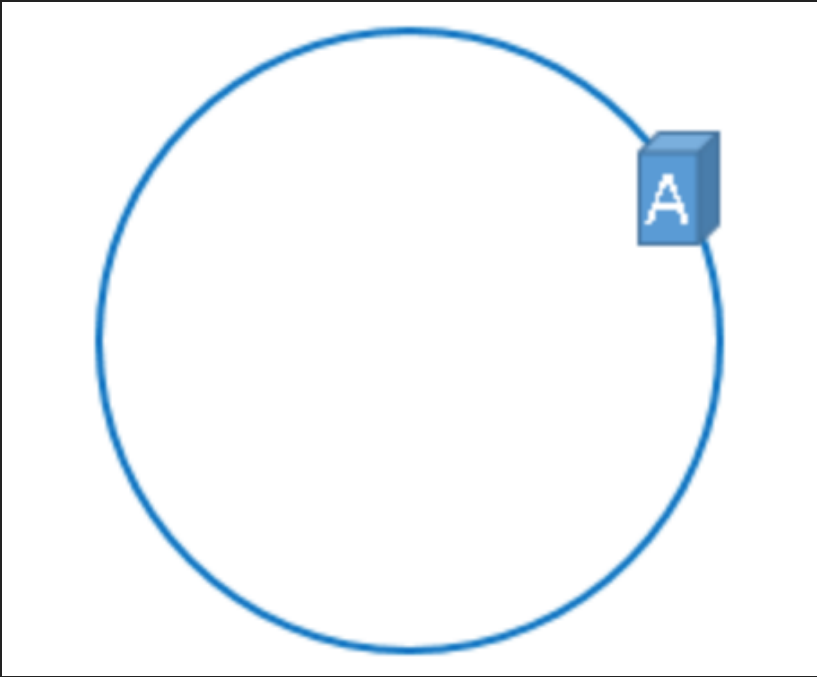
- ▶ 平衡性
 - ▶ key 能够均匀分布到不同bucket
 - ▶ “虚拟节点”
- ▶ 单调性
 - ▶ 一个bucket故障，引起所有bucket不可用
 - ▶ bucket增删尽量不影响原有的kv映射

支持多bucket S3存储驱动实现

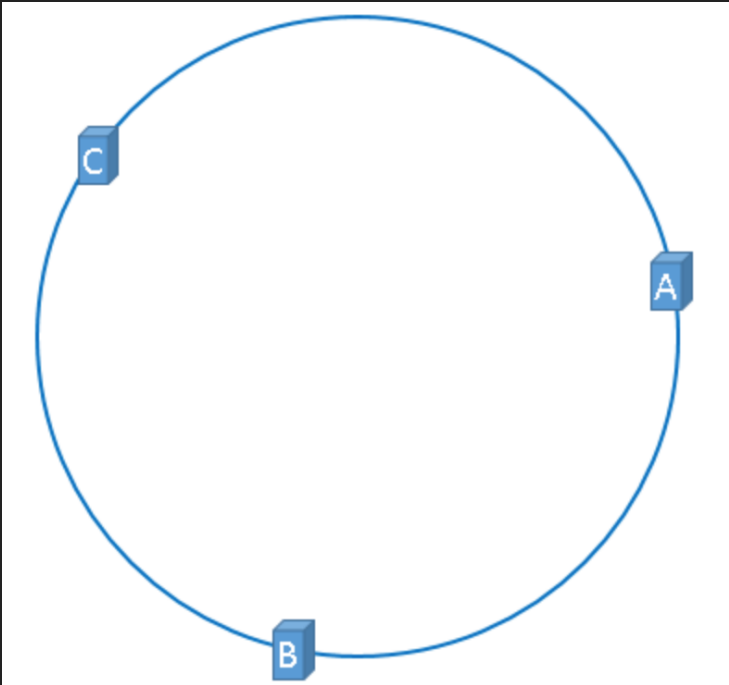
一致性哈希算法



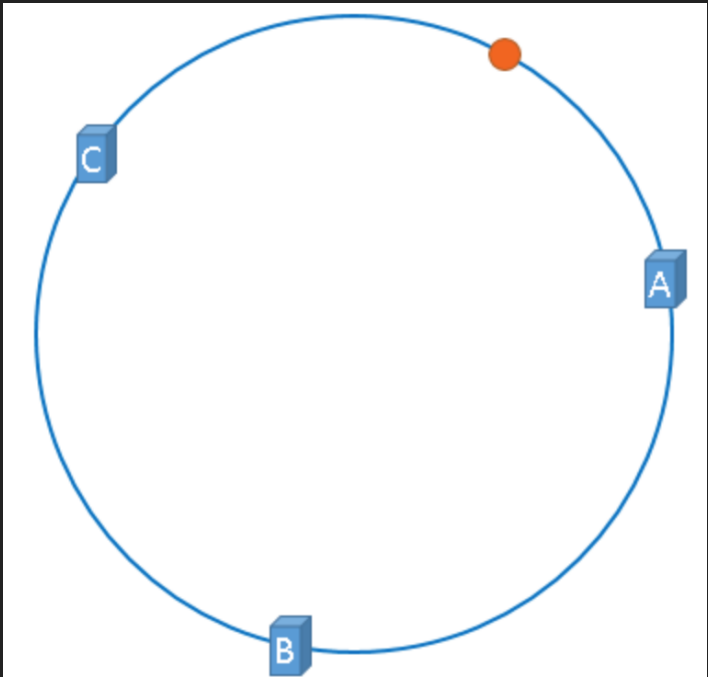
2^{32}



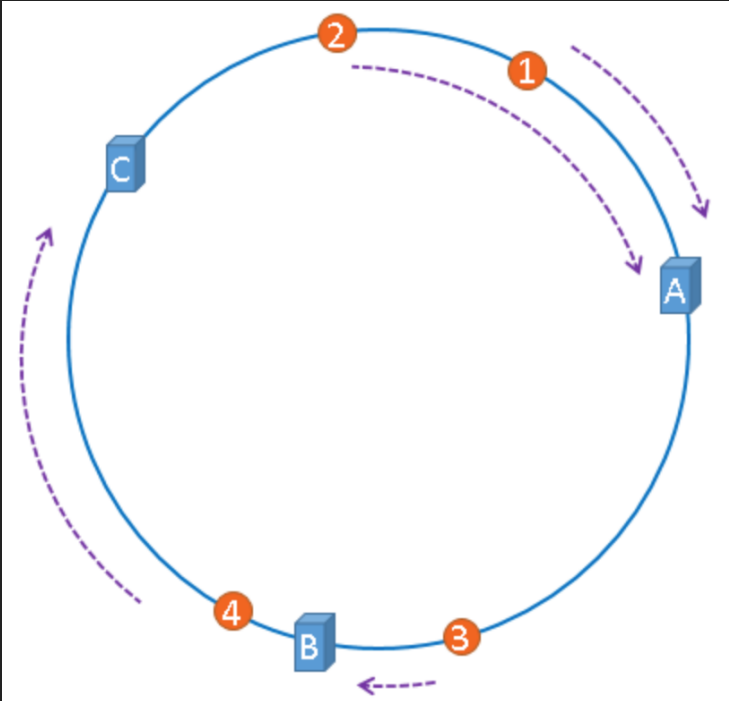
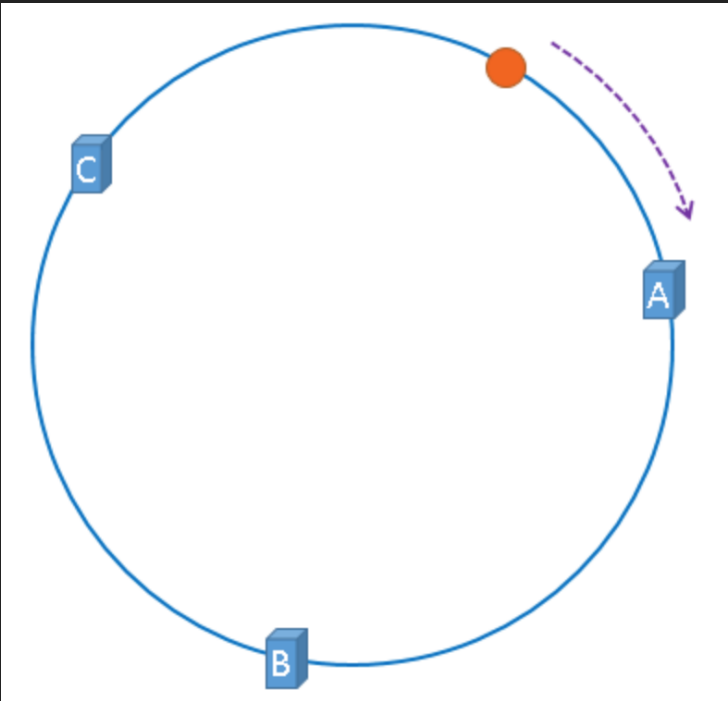
$\text{hash}(\text{bucketA}) \% 2^{32}$



$\text{hash}(\text{bucketB}) \% 2^{32}$
 $\text{hash}(\text{bucketC}) \% 2^{32}$

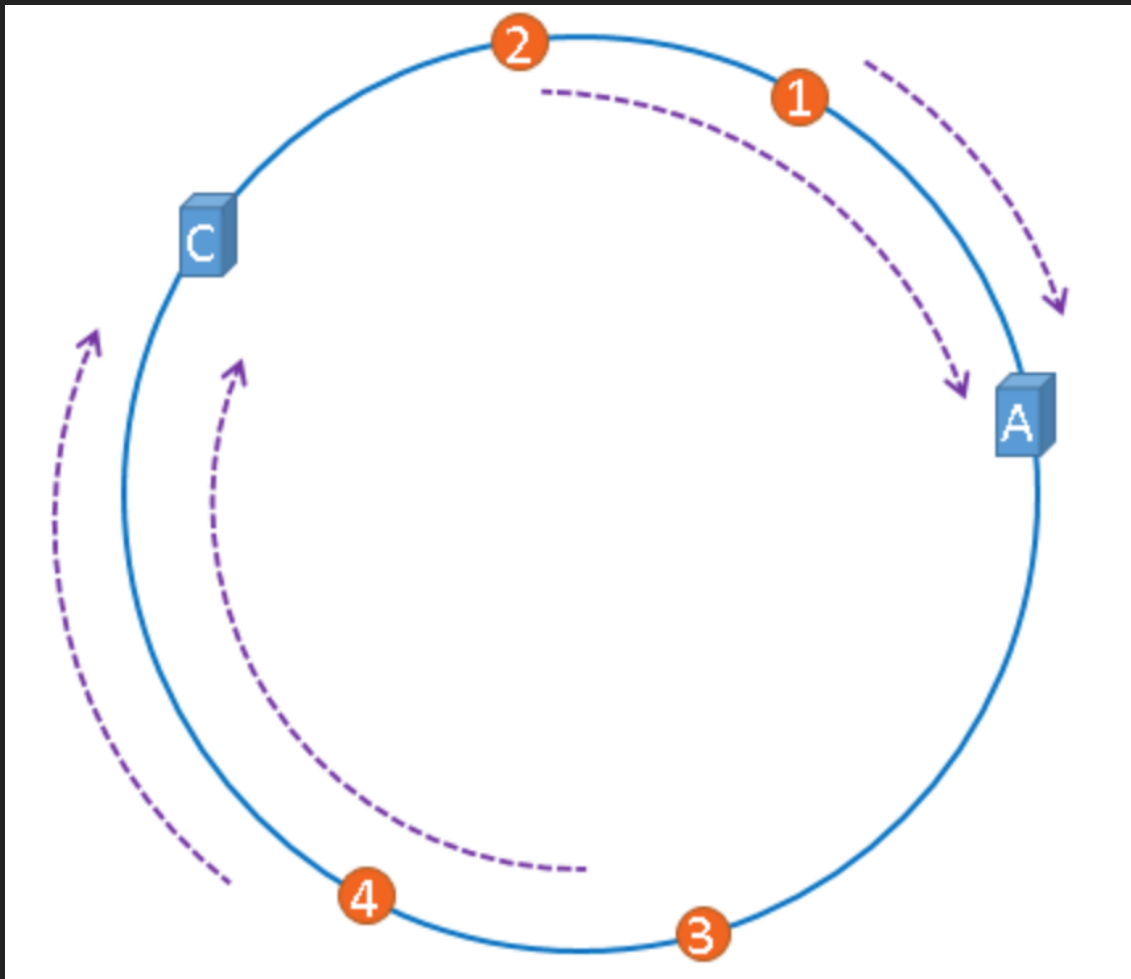


$\text{hash}(\text{digest}) \% 2^{32}$

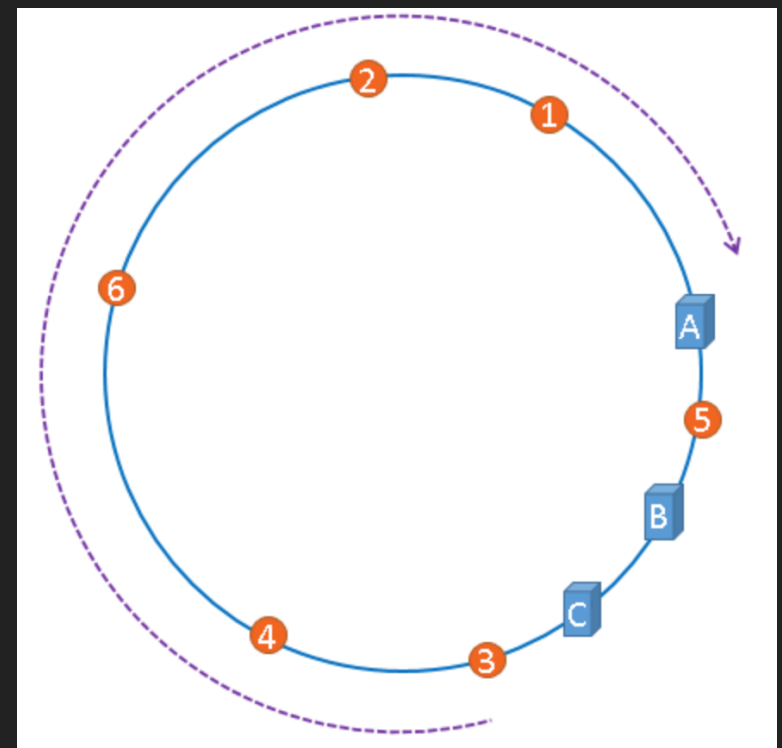
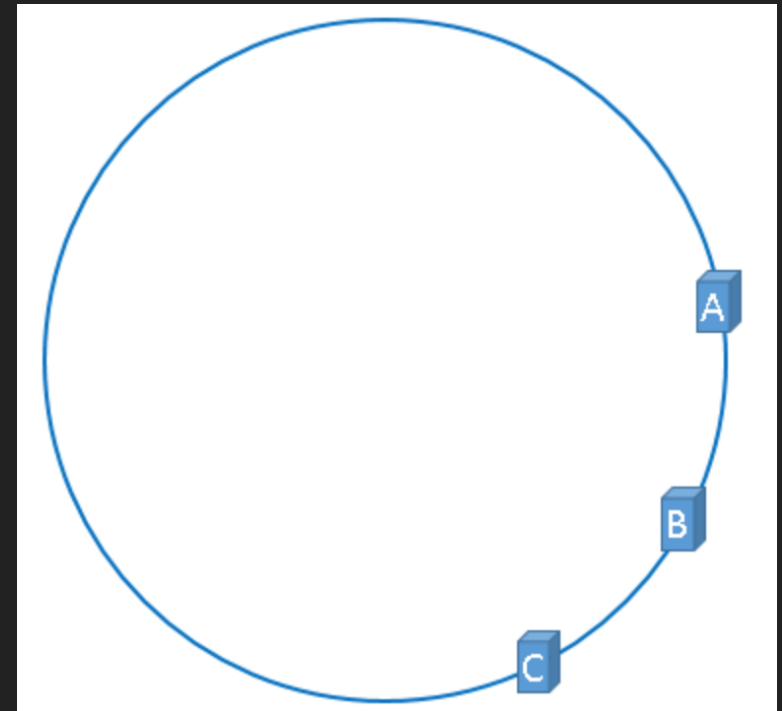


支持多bucket S3存储驱动实现

一致性哈希算法优点



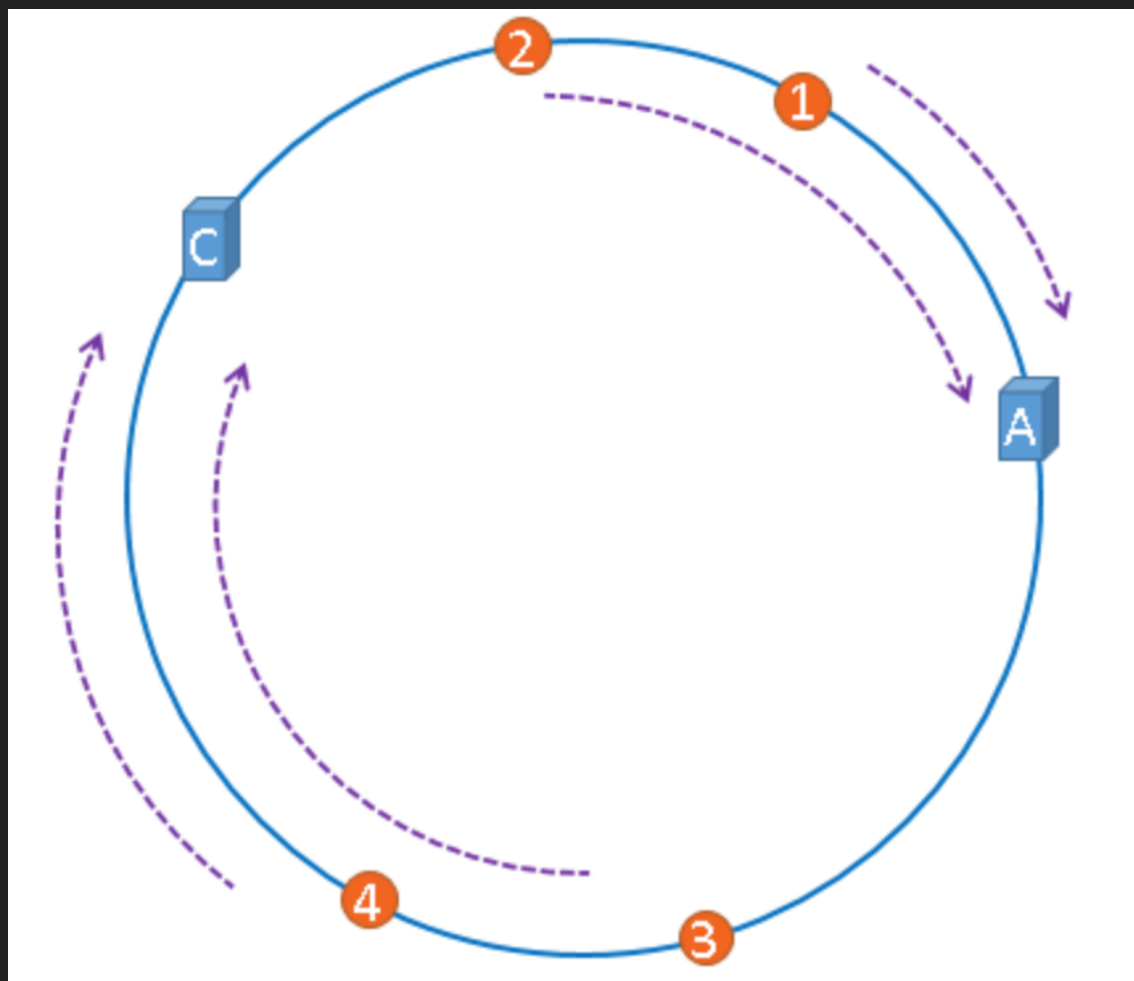
分布式多bucket故障场景



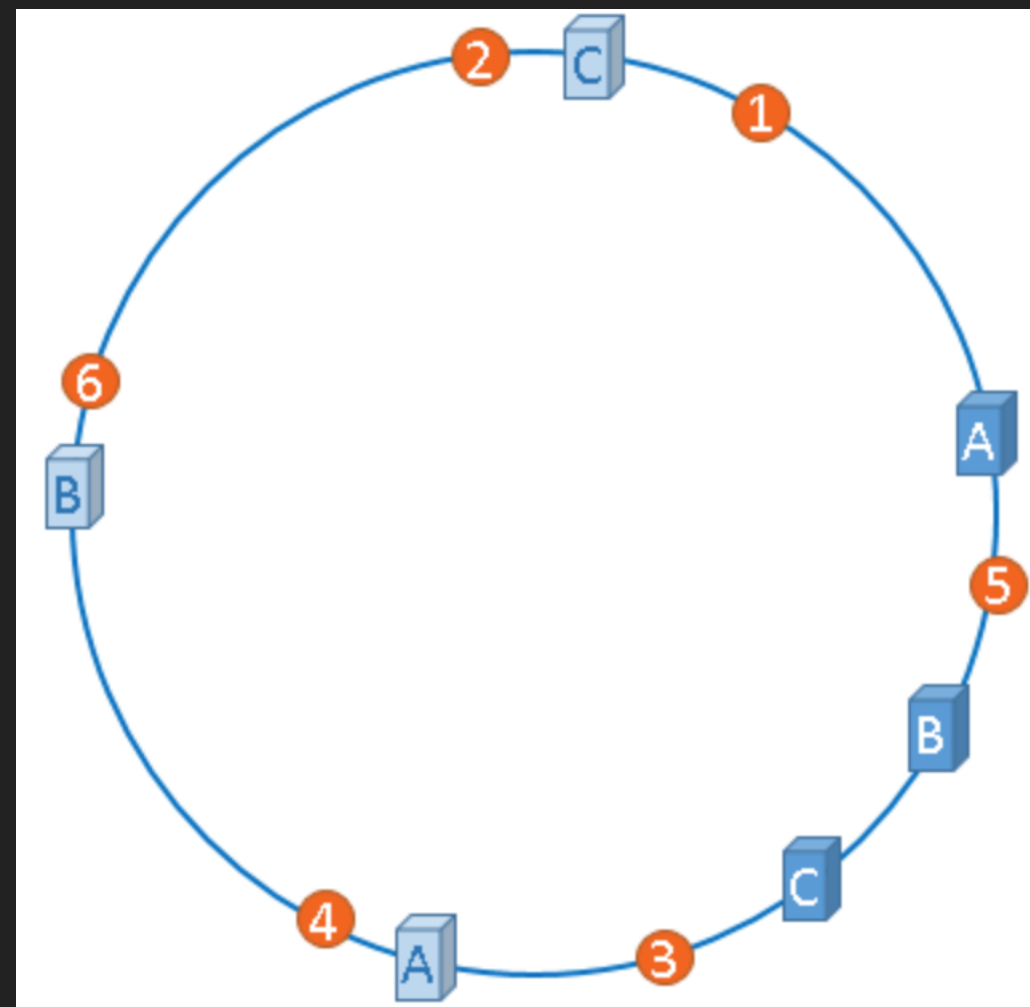
虚拟节点

支持多bucket S3存储驱动实现

一致性哈希算法优点



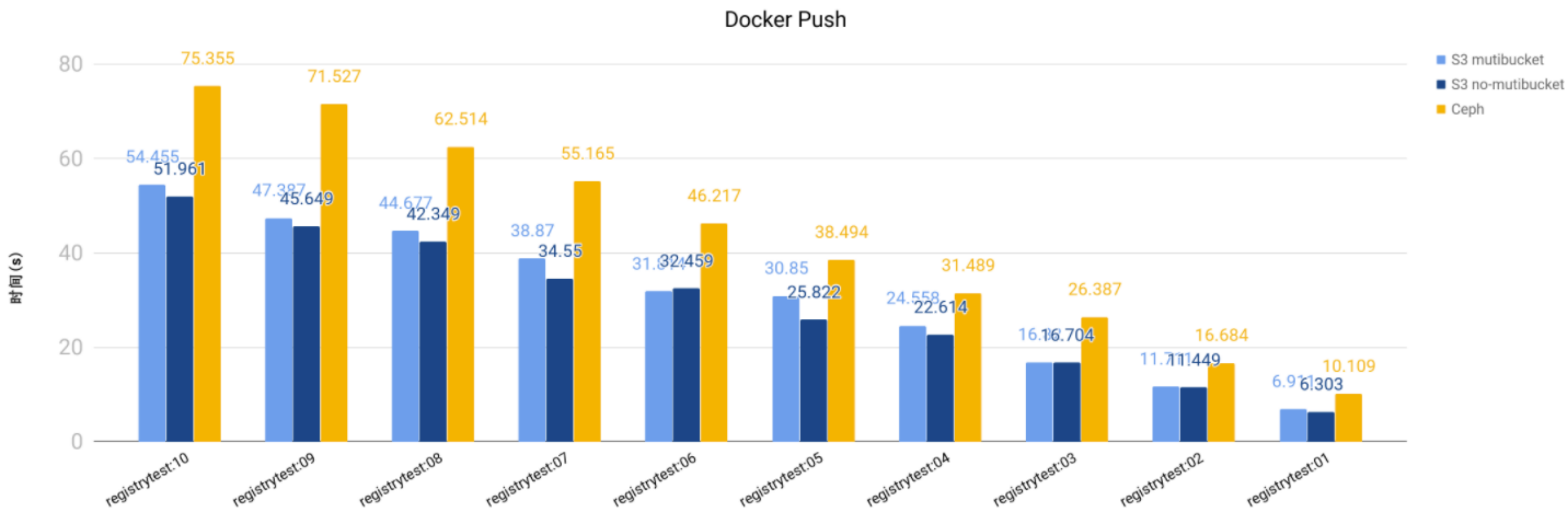
分布式多bucket故障场景



虚拟节点

性能测试对比

对比多bucket, 单bucket, Ceph rados 对上传镜像速度的影响

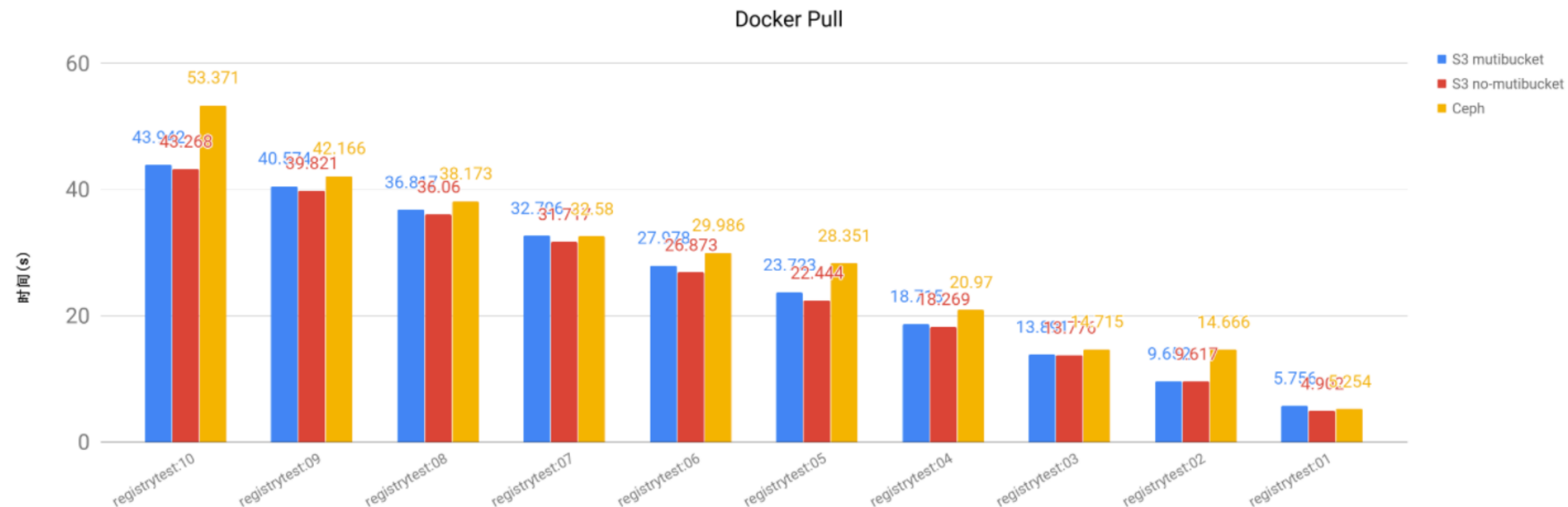


结论:

- 使用v2.5.2 distribution比v2.1.1上传性能显著增强

性能测试对比

对比多bucket, 单bucket, Ceph rados 对下载镜像速度的影响

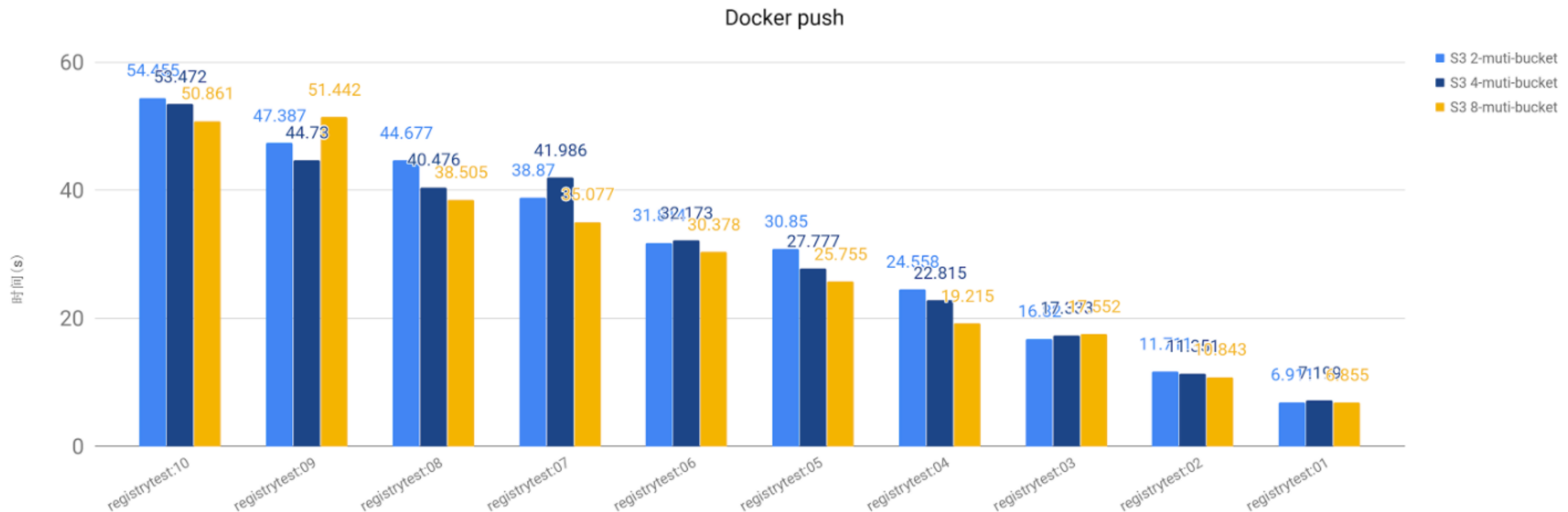


结论:

- 使用v2.5.2 distribution比v2.1.1下载性能显著增强

性能测试对比

bucket数量对上传速度的影响

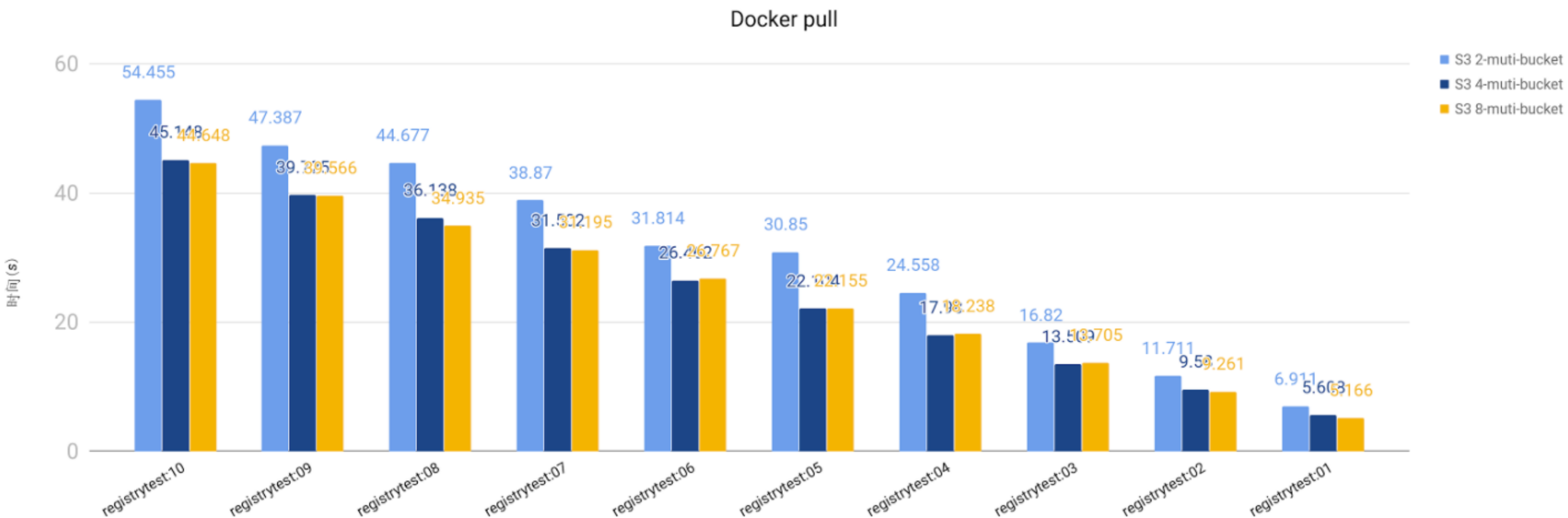


结论:

- 使用多个bucket的场景对上传速度影响不大。
- 单个bucket故障时, 减小故障影响面, 加快恢复速度。

性能测试对比

bucket数量对下载速度的影响



结论:

- 使用多个bucket的场景对下载速度影响不大。
- 单个bucket故障时, 减小故障影响面, 加快恢复速度。

总结遇到的问题

- ▶ Registry源码编译
 - ▶ golang.org / google.com 等无法访问，翻墙
 - ▶ glide mirror/ base
 - ▶ github.com/sock-cli 解决，使用sock2http
- ▶ go aws library未使用基于DNS解析的bucket路由
 - ▶ 小小吉S3不支持（path-style）格式的资源访问
 - ▶ secure参数的坑：yes = https , no = http
 - ▶ v4auth：官方说默认关闭
 - ▶ 实际上在registry v2.4.1 默认使用v4auth
 - ▶ 大量配置并未实现

总结遇到的问题

- ▶ 两个S3 driver s3-aws / s3-goamz
 - ▶ 使用RegionEndpoint时, 不去检验Region
 - ▶ 在v2.6.2实现了S3 V2auth的兼容, 生成证书 setv2handler(s3obj)