华为Git实践

工作模式创新，及多中心分布式架构

蒋 鑫

华为技术有限公司

北京　伦敦　纽约　旧金山　圣保罗　上海　东京

QCon 全球软件开发大会

[上海站]

信息安全　机器学习　人工智能　黑产　互联网金融（FinTech）　团队管理　基础设施　云计算　软件性能　硅谷　微服务　互联网架构

主办方 Geekbang 极客邦科技　InfoQ

2017年10月17-19日
上海·宝华万豪酒店

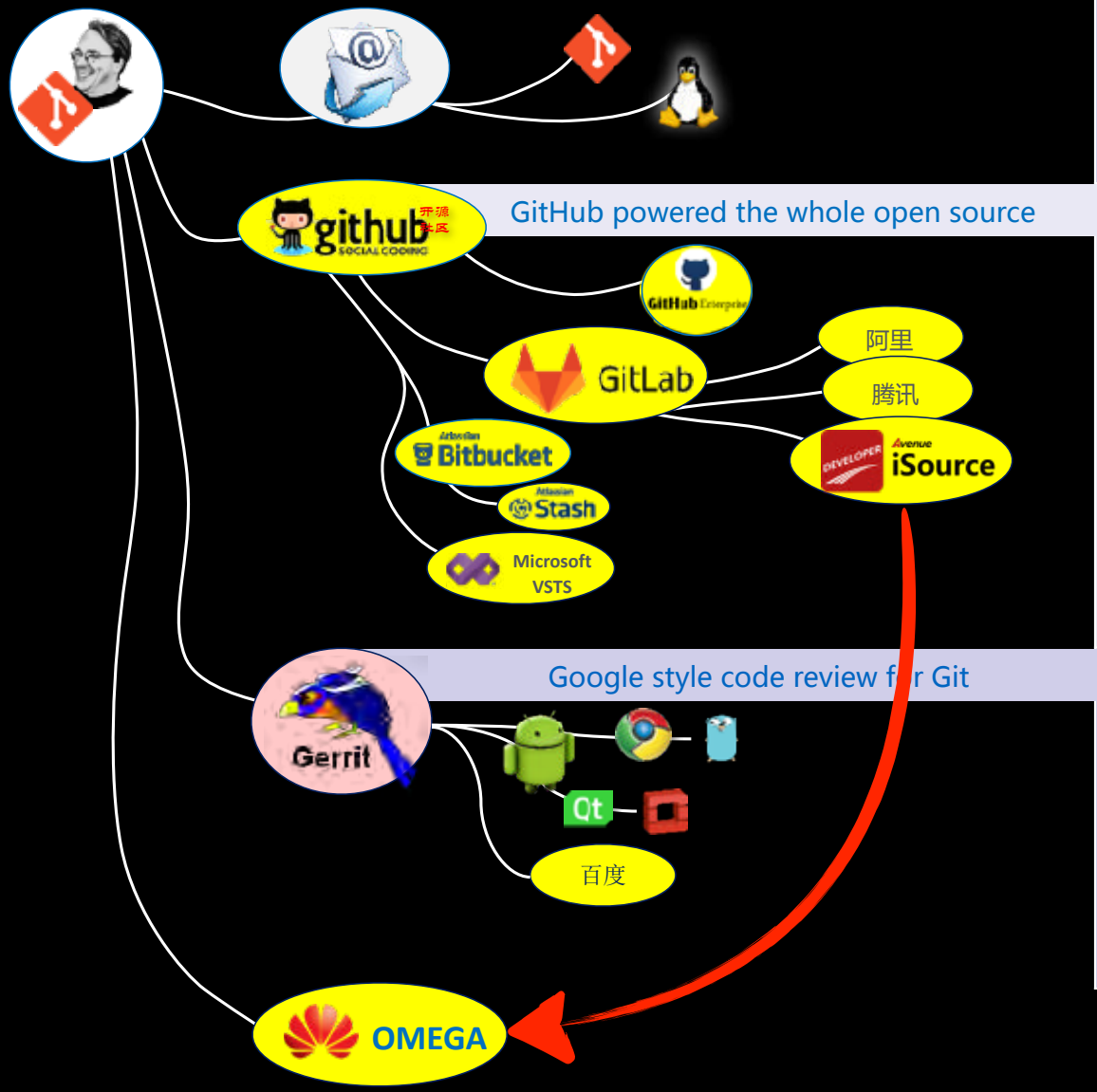
扫描二维码
开启软件开发新思路

蒋鑫

微博 / 微信：gotgit， http://www.worldhello.net

- Git contributor (git clean -i, ...)
- Git l10n coordinator (https://github.com/git-l10n/git-po)
- 《Git权威指南》，2011 (https://github.com/gotgit/gotgit)
- 2005—2015，自雇佣，前一半时间推广SVN，后一半时间在纠正这个错误
- 2015.12 加入华为 iSource 团队
  - “买买买？还是自研？”

Git Family



- Distributed; freedom; commit any where, any time
- Geeky

- Fork + Pull Request
- Social coding
- UI

- Sync of fork repos; waste of disk spaces
- Multi-repo management, git submodule?

- Review in Google style
- Multi-repo solution: repo

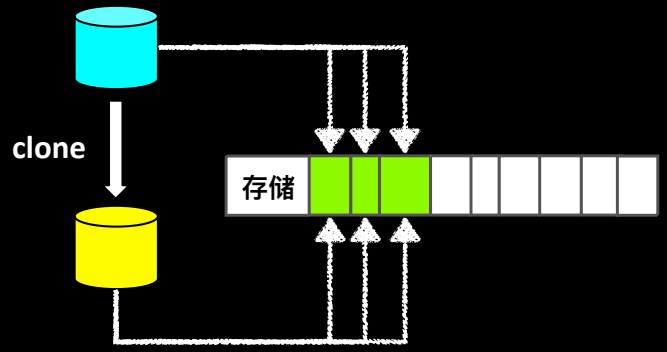
- Project by project, not a platform
- Centralized, lack of freedom
- Just for review, and the UI

TABLE OF
CONTENTS 大纲

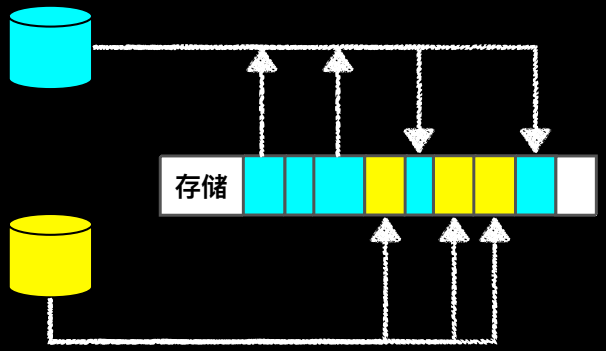
- Git工作流创新
- 多中心Git架构

分布式工作流的困扰1：服务器端的存储压力

Fresh clone, and hard links



After new push and GC

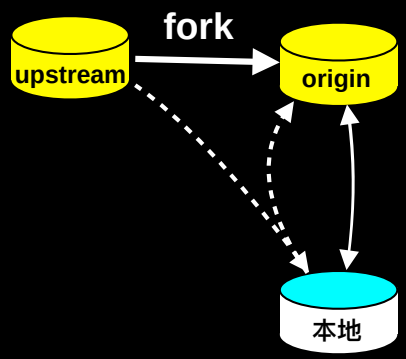



```
--shared, -s
    When the repository to clone is on the local machine, instead of using hard links,
    automatically setup .git/objects/info/alternates to share the objects with the source
    repository. The resulting repository starts out without any object of its own.

    NOTE: this is a possibly dangerous operation; do not use it unless you understand what it
    does. If you clone your repository using this option and then delete branches (or use any
    other Git command that makes any existing commit unreferenced) in the source repository,
    some objects may become unreferenced (or dangling). These objects may be removed by
    normal Git operations (such as git commit) which automatically call git gc --auto. (See
    git-gc(1).) If these objects are removed and were referenced by the cloned repository,
    then the cloned repository will become corrupt.
```

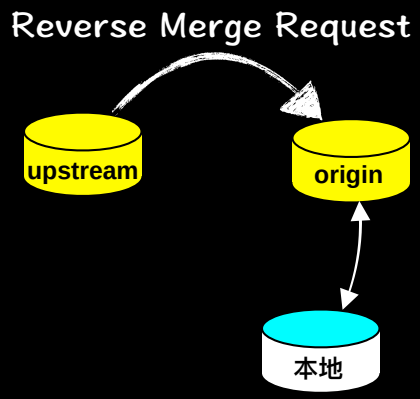
分布式工作流的困扰2：客户端操作的复杂性

Boring procedure

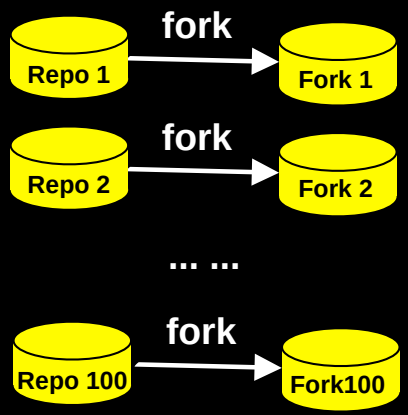



```
git remote add upstream URL
git fetch upstream
git rebase upstream/master
git push -f origin HEAD
```

Strange procedure



How about 100 repos?



How about topic branches?

- slow down clone/fetch
- loose references，high IO
- chaos of visibility of all refs

分布式工作流的困扰3：多仓库关联

- Multiple repos in one project, why?
  - GC, clone are quite slow for big repos; auth; micro services
- git submodule：适用范围：仓库少，松耦合
  - Recurisive submodules: 描述复杂的目录嵌套
  - Update of gitlinks. 迟更新和超前更新问题
  - Conflict of gitlinks. 支持几万研发，简直是灾难
- 2016年，iSource下的一个项目解耦，20子仓扩增到1000子仓... ...

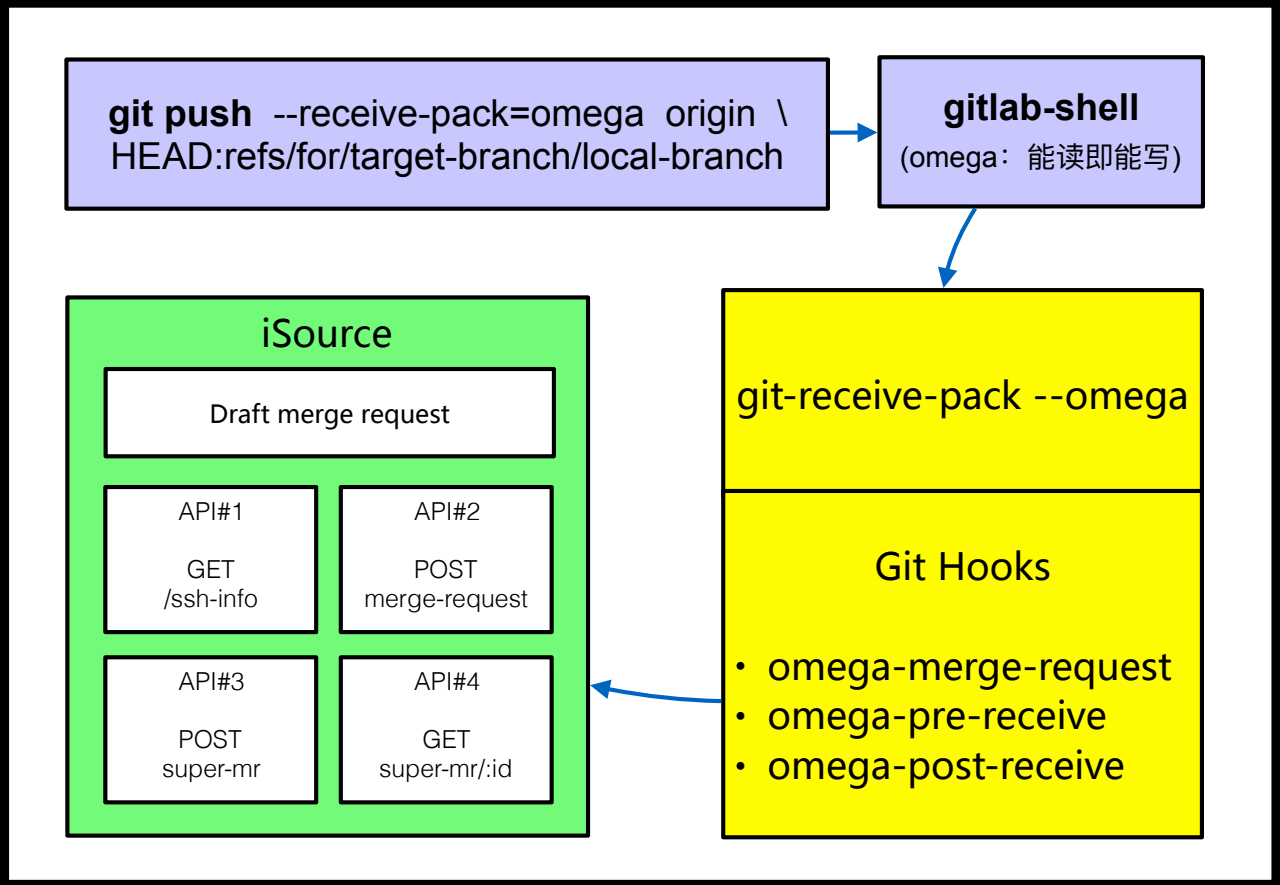
No fork,

no feature branch,

and no submodules

OMEGA（支持集中式工作流）

One-stop Multi-Endpoints Git Access



- No fork
  - Create PR/MR by push
  - Draft mode PR/MR
- No feature branch
  - Special referenes: refs/merges/123/head
- No submodules
  - Manifest repo and XML
  - git-mm: rewrite repo in golang for OMEGA

OMEGA模式操作示例

git push origin HEAD:refs/for/master/my/topic1

```
对象计数中：3，完成。
Delta compression using up to 2 threads.
压缩对象中：100% (2/2)，完成。
写入对象中：100% (3/3)，415 bytes | 0 bytes/s，完成。
Total 3 (delta 0), reused 0 (delta 0)
remote: ################################################################################
remote: #                                                                              #
remote: #  New merge request created at: refs/merges/1/head                           #
remote: #                                                                              #
remote: ################################################################################
To [redacted]
 * [new branch]      HEAD -> refs/for/master/my/topic1
```

 # more git commit ...
git push origin HEAD:refs/for/master/my/topic1

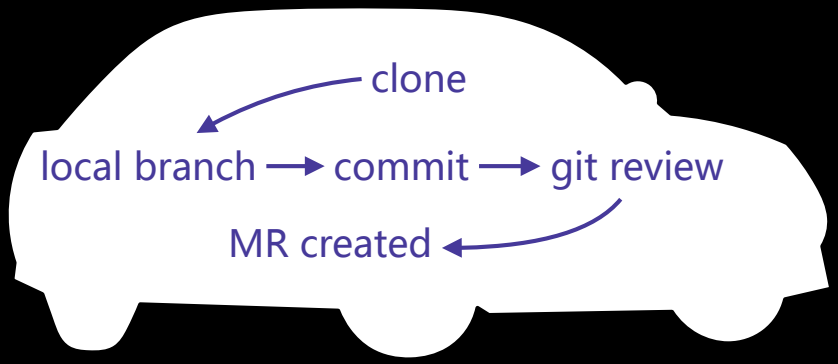
```
对象计数中：3，完成。
Delta compression using up to 2 threads.
压缩对象中：100% (2/2)，完成。
写入对象中：100% (3/3)，419 bytes | 0 bytes/s，完成。
Total 3 (delta 0), reused 0 (delta 0)
remote: ################################################################################
remote: #                                                                              #
remote: #  Merge request updated at: refs/merges/1/head (was b7605d1c)                 #
remote: #                                                                              #
remote: ################################################################################
To [redacted]
 * [new branch]      HEAD -> refs/for/master/my/topic1
```

git push origin HEAD:refs/for/master/my/topic2

```
Total 0 (delta 0), reused 0 (delta 0)
remote: ################################################################################
remote: #                                                                              #
remote: #  New merge request created at: refs/merges/2/head                           #
remote: #                                                                              #
remote: ################################################################################
To [redacted]
 * [new branch]      HEAD -> refs/for/master/my/topic2
```

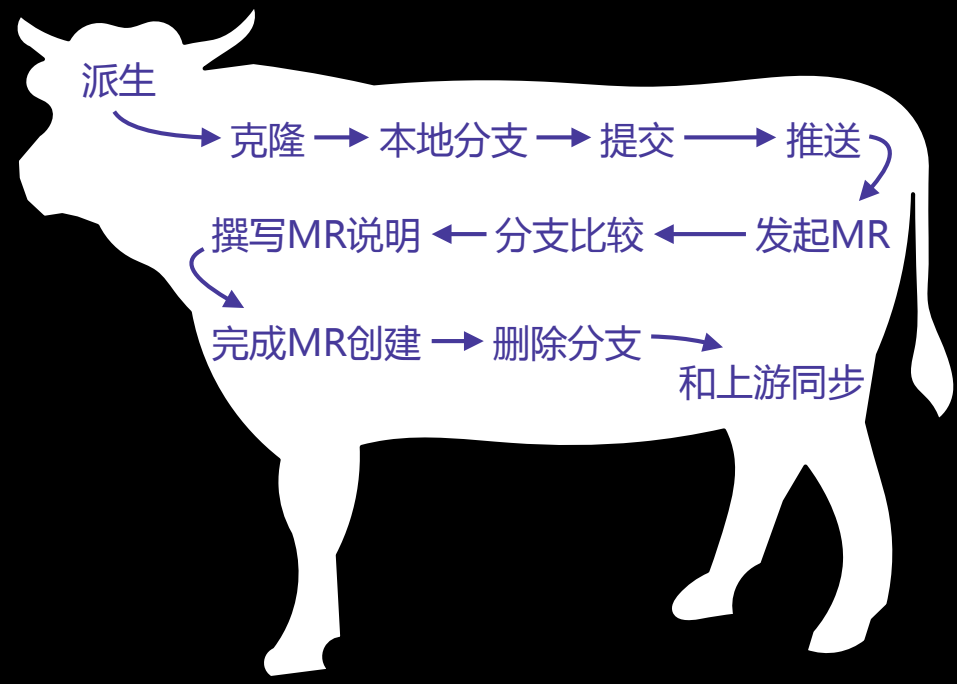
两种工作流的对比

集中式工作流



VS

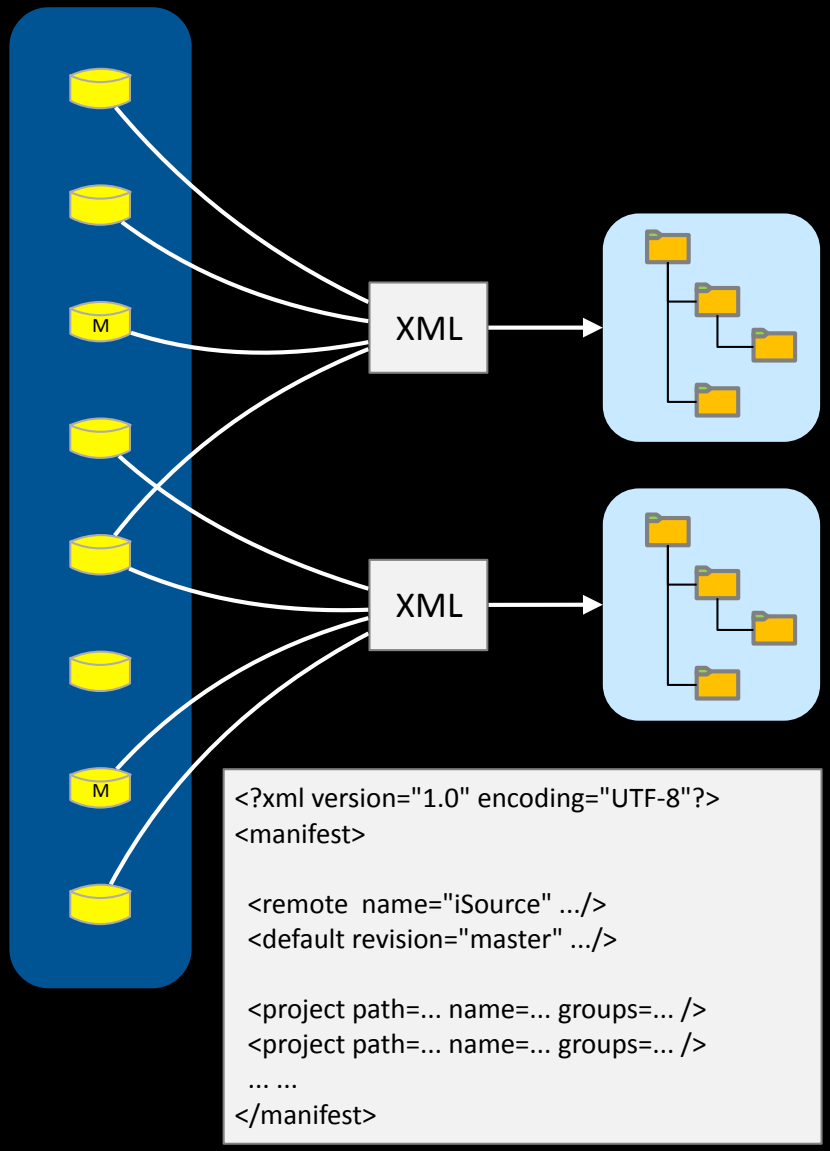
典型分布式工作流



Gerrit与OMEGA

	Gerrit	OMEGA
服务名	--receive-pack="gerrit receive-pack [options]"	--receive-pack="omega [options]"
代码检视	Change (per commit)	Merge Request (per feature)
特殊refspec	HEAD:refs/for/master HEAD:refs/drafts/master	HEAD:refs/for/master/local/branch HEAD:refs/drafts/master/local/branch
提交和检视任务的关联	• git hook: commit-msg • 在提交说明中嵌入唯一的标识 • Change-Id: I... ...	• No client side hooks • 无须在提交说明中嵌入特殊标识 • 同一个用户 / 本地分支对应同一个MR
多仓库管理模式	Manifest	
客户端工具	repo • Only for Linux • Only support multiple repos • Clone from bundle	git-mm • 支持 Linux、Windows 等 • 既支持多仓模式，也支持单仓操作 • 支持 git-lfs • 支持 bundle * • 支持 batch query *

多仓库管理工具 git-mm

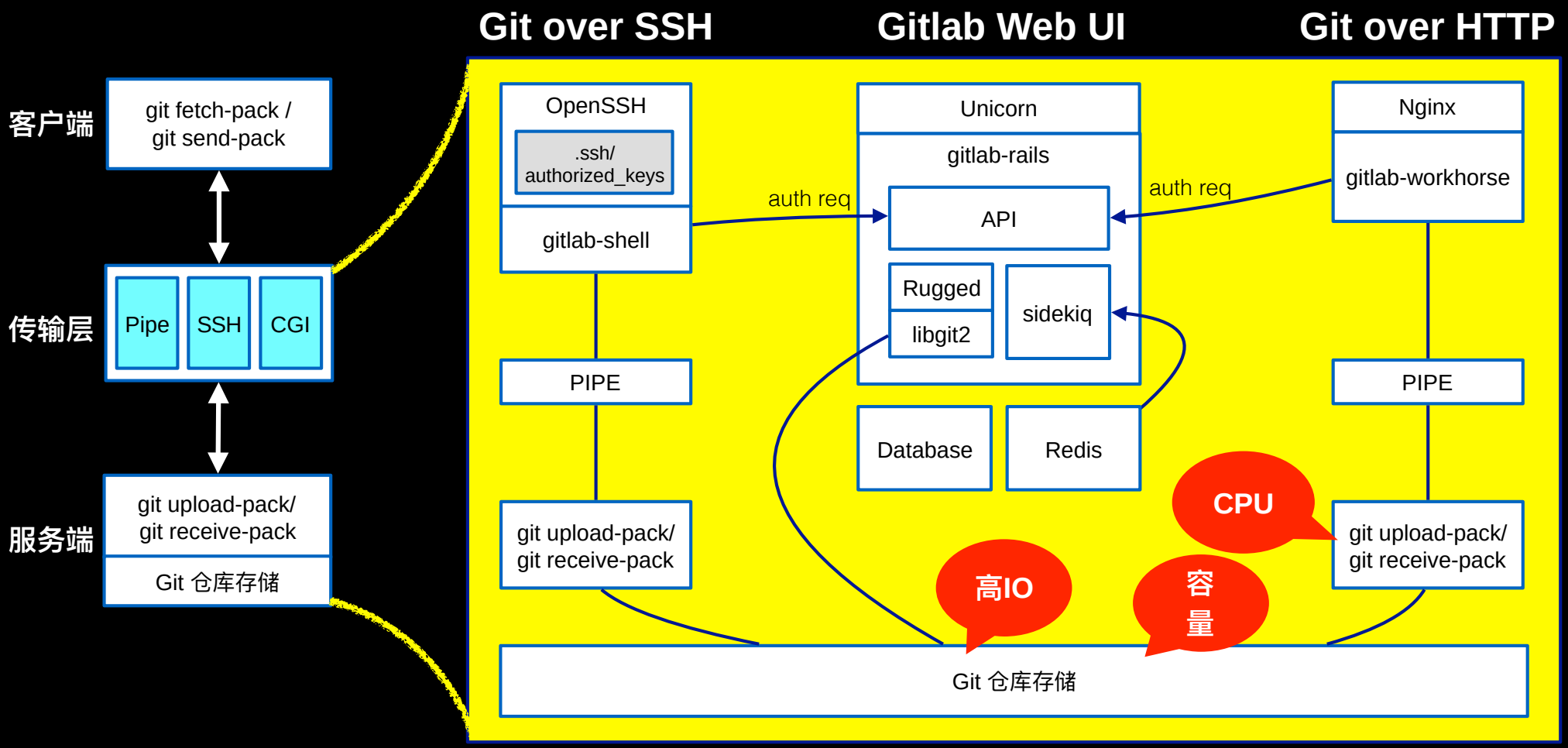


- Git-MM： Git Multi-Module，参考Android repo方案，是适配 OMEGA 的 repo，用 Golang开发。
- 提供统一的Git环境设置。参考：Autodesk/enterprise-config-for-git
- 同时提供多仓和单仓解决方案。
  - git config --global alias.review “git-mm --single”
- Bundle 和 batch request
- 示例
  - git mm init -u URL
  - git mm sync
  - git mm upload
  - git review

TABLE OF
CONTENTS 大纲

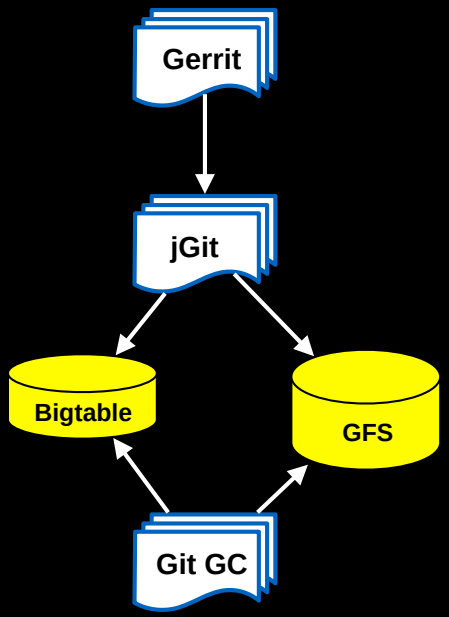
- Git工作流创新
- 多中心Git架构

Gitlab (<8.10) 架构

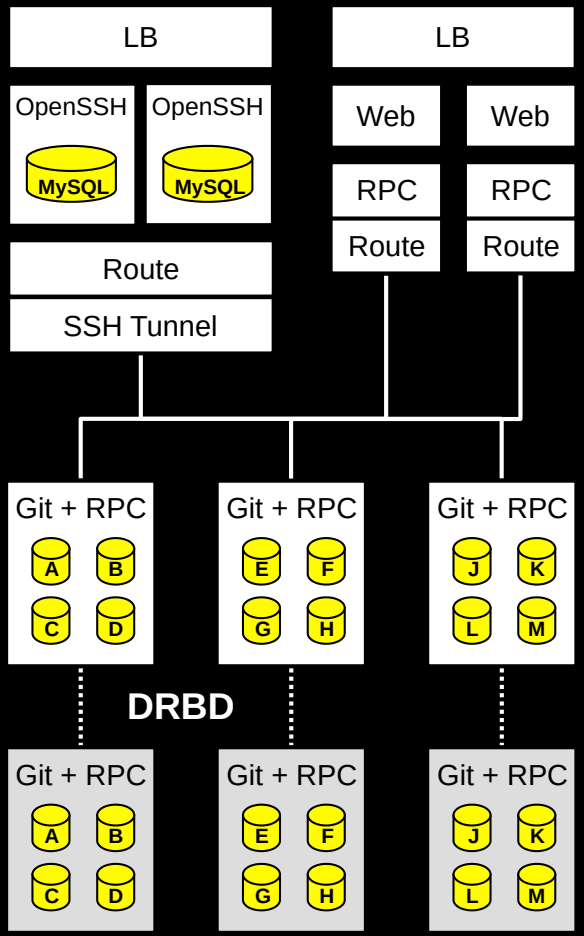


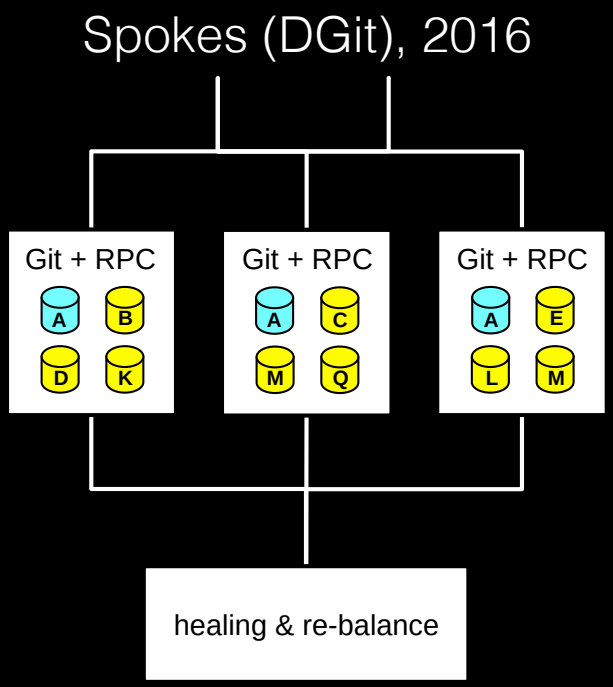
业界方案

Google
分布式存储方案

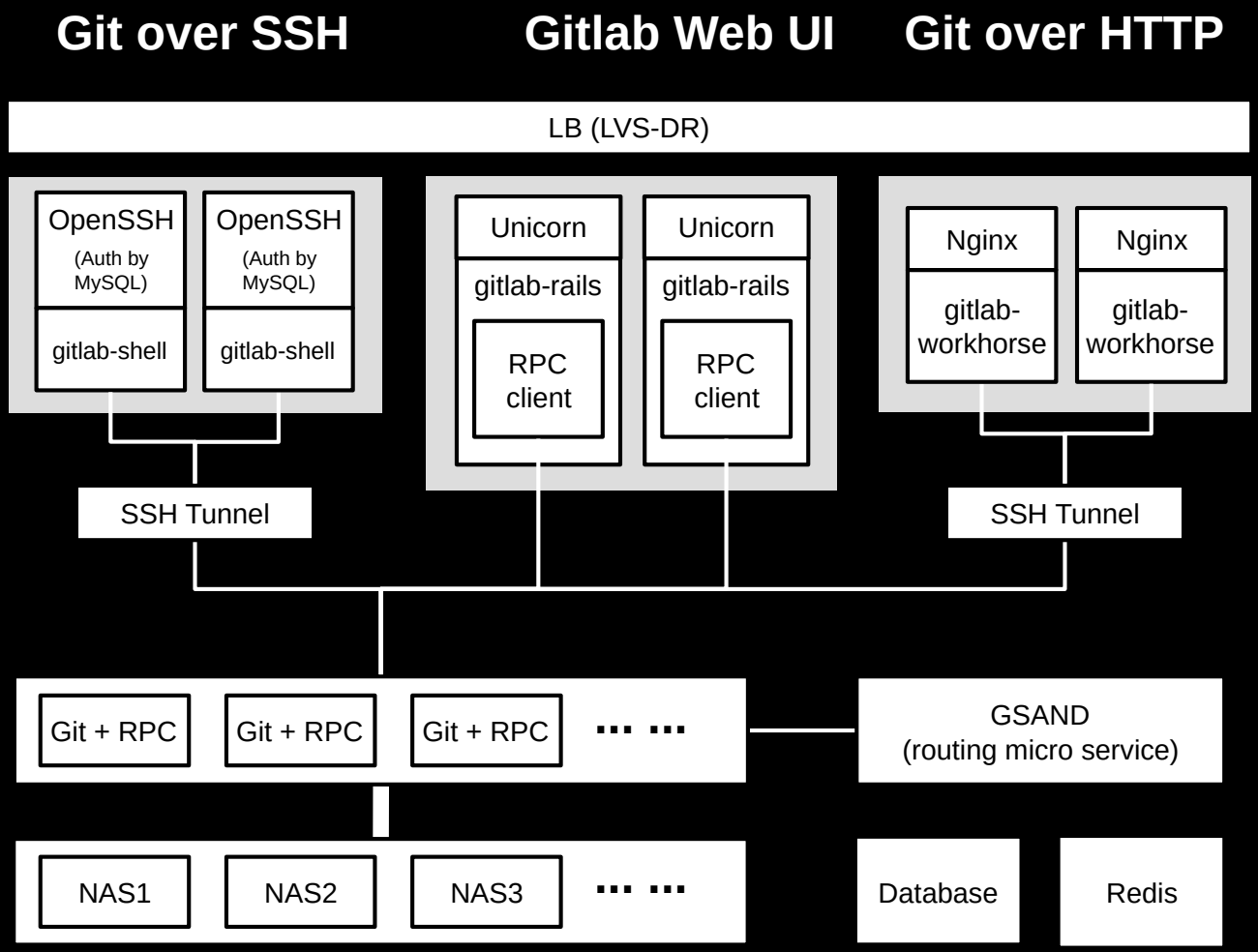


GitHub
本地存储分片方案





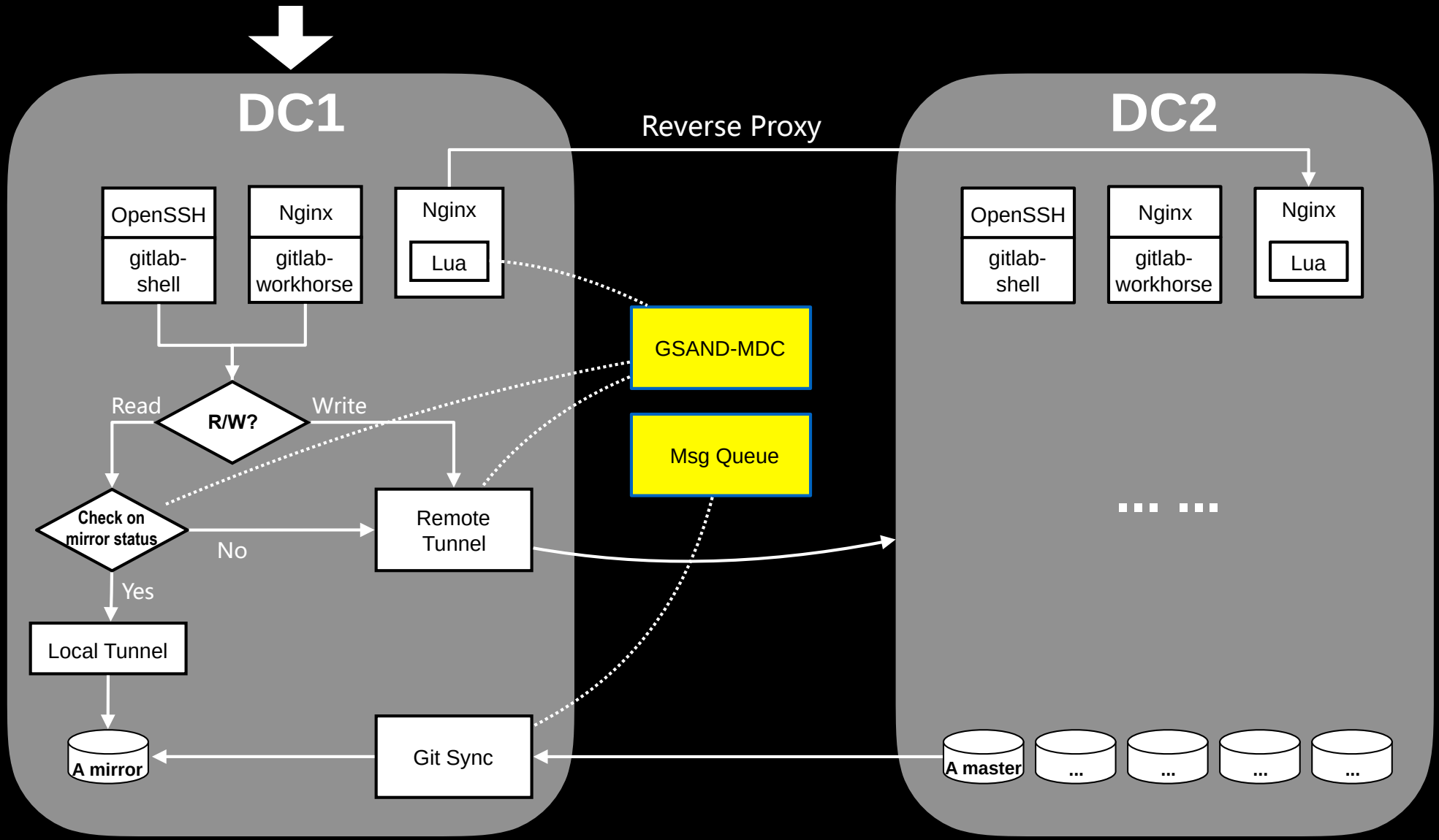
iSource 数据中心内集群方案



TODO

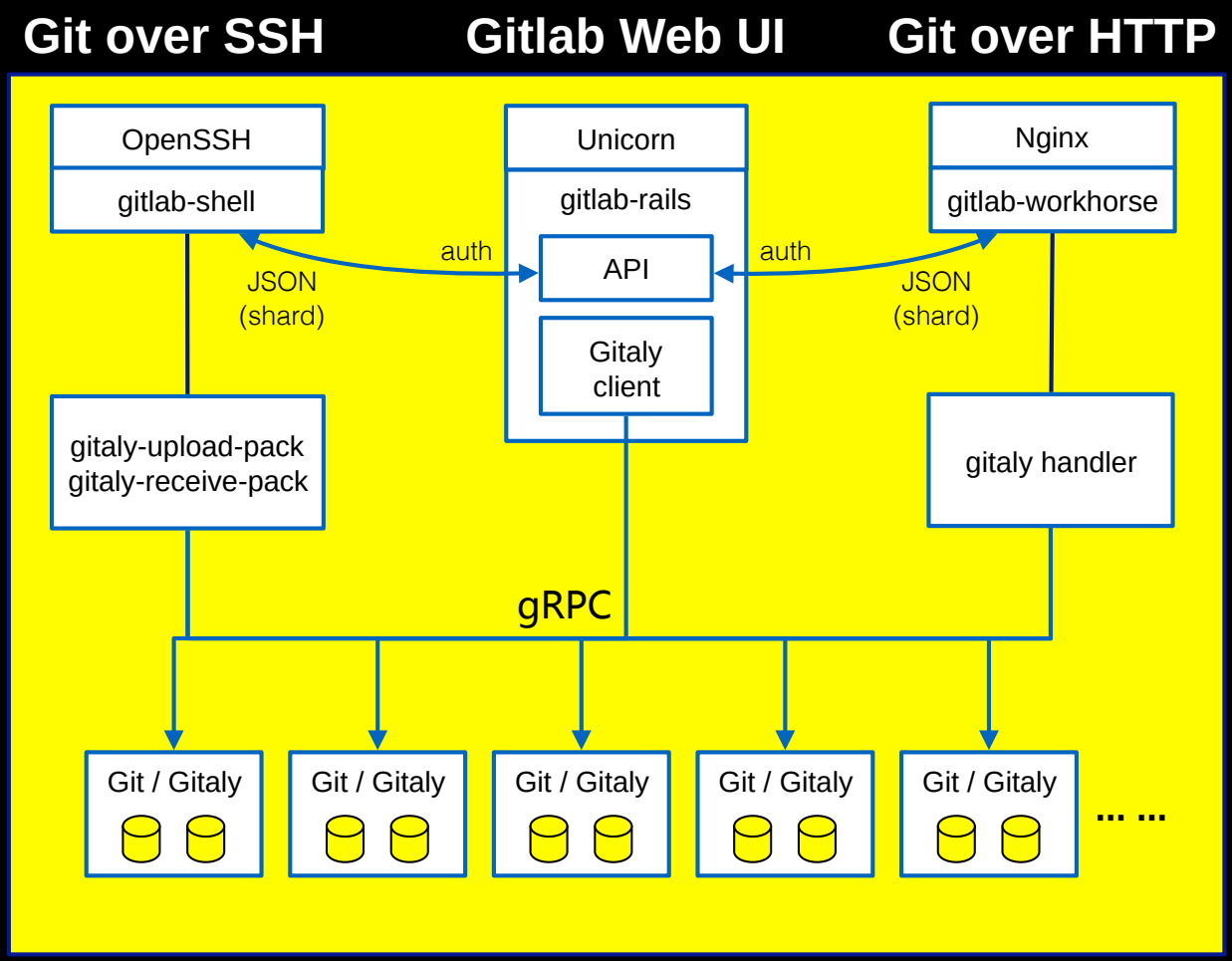
- Git靠近存储，无NFS
- 应用层三副本复制

iSource 多数据中心方案



Gitlab 社区版本的架构演进

- 《How We Knew It Was Time to Leave the Cloud》
- 《Why we are not leaving the cloud》
- “don’t want to be an infrastructure company”
- “using the simplest, most boring solution for a problem”



小结

- 我们喜欢Pull Request，喜欢自由地创建项目，更喜欢集中式带来的便利。我们痛恨submodule，喜欢 git-mm。
- 面对每日超过 xxx 万次的Git访问、xx TB以上的流量，我们设计并运维着多数据中心Gitlab集群方案，支撑了高并发、跨地域的协同。
- 参考资料：
  - Git at Google: Making Big Projects (and Everyone Else) Happy, Dave Borowitz - Git Merge 2015
  - How We Made GitHub Fast (2009)
  - Introducing DGit (2016)
  - Building resilience in Spokes (2016)

THANKS!

让创新技术推动社会进步

HELP TO BUILD A BETTER SOCIETY WITH
INNOVATIVE TECHNOLOGIES

Geekbang.
极客邦科技

InfoQ

专注中高端技术人员的技术媒体

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

高端技术人员学习型社交平台

StuQ 斯达克学院

实践驱动的 IT 教育平台

地址：北京市朝阳区洛娃大厦C座8层1801室　　网址：www.geekbang.org