DevOps进化之路

从运维到开发 一路走来的经验分享

范余乐

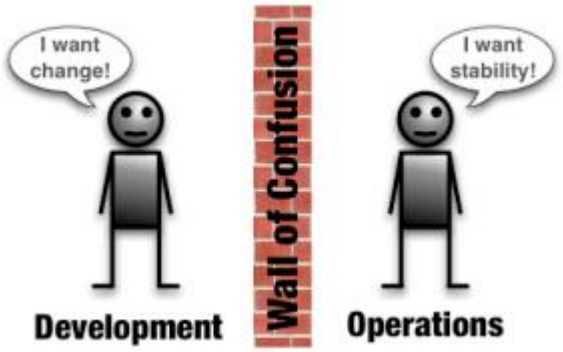
2014-10-18

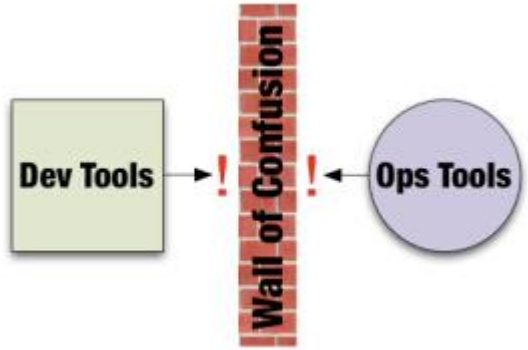
提 纲

- DevOps简介
- DevOps实践
- 总结

什么是DevOps

- Development & Operations
- 传统模式 : 运维——开发





- 运维需要做的改变: 人工、经验 -> 脚本、工具 -> 平台、产品
- DevOps =
  - 将运维的事情系统化,平台化
  - 方便开发人员; 解放运维人员
  - 有助于企业实现 “业务敏捷性” 和 “IT融合”

怎么做到DevOps

获取需求

- 梳理并统一标准化的流程
- 基础设施标准化
- 日常遇到的某一类需求
- 用户是谁

自动化工具支持

- 代码管理
- 配置管理
- 批量部署工具
- ……

Web化形成平台

- 运维人员思路转变
- 学习一点html基础
- 学习一个web框架

DevOps实践——从脚本开始

- 一个代码发布脚本
  - 代码从哪里来?
  - 代码怎么发布?
  - 发布完要干啥?

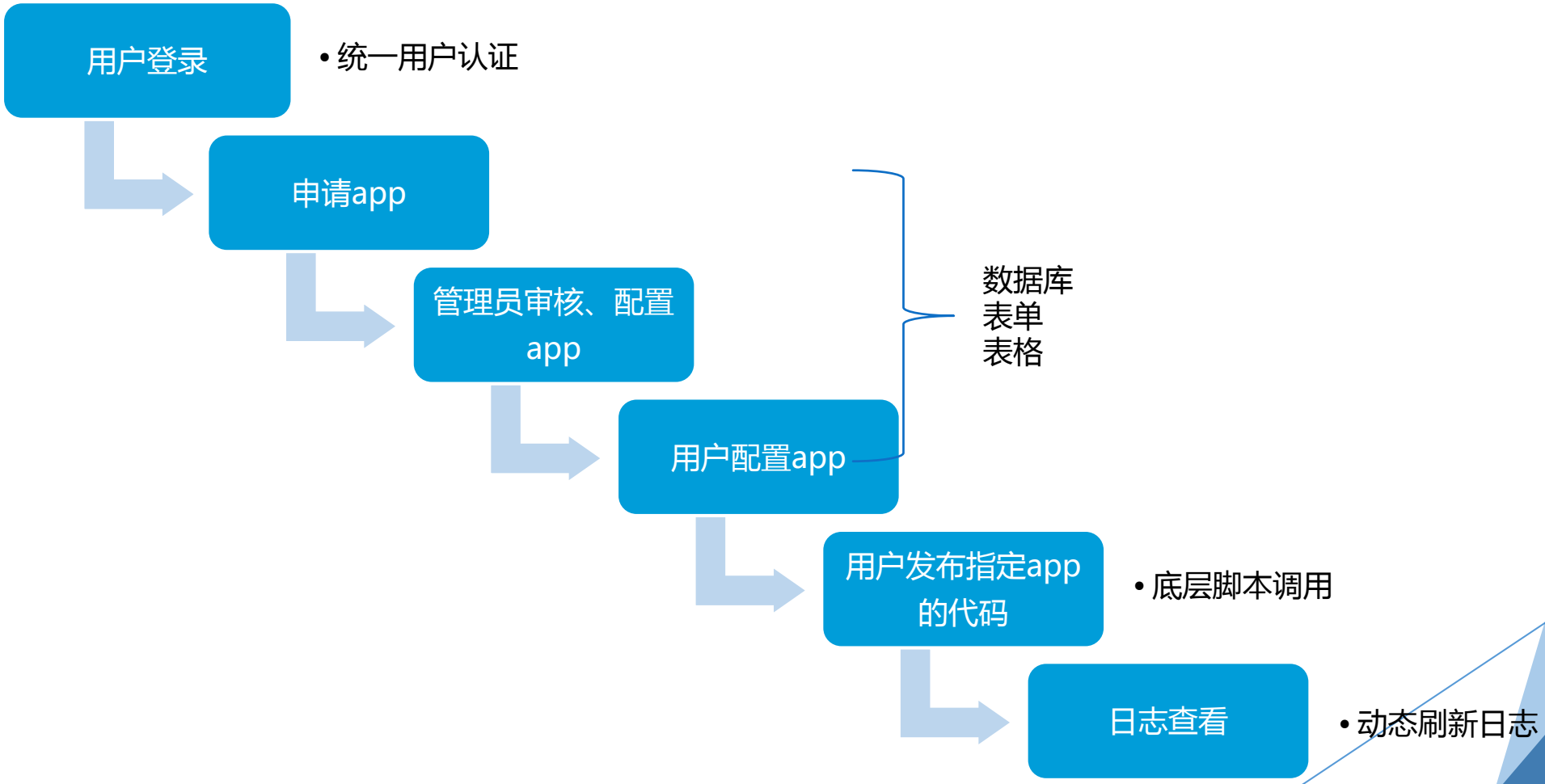
```
[ 09:59:32-root@fzdm-10-59-95-4:Queue ]#./codeupdate.py -h
./codeupdate.py -h
Usage: codeupdate.py [options]

Options:
  -h, --help  show this help message and exit
  -H HOST     Host list
  -u URL      Svn url
  -t PATH     Target path
  -i APPID    AppID
  -e PATH     Exclude path
  --cu=USER   Code user
  --ru=USER   Run user
  --rd=PATH   Path need write permission
  -v VER      Svn version
  -T STRING   Name of log title
```

- 类似的开源项目： Jenkins、 BuildBot

代码发布系统 —— 需求分析

易用的用户界面	• Web / GUI • 支持用户认证
支持代码的发布/回滚	• 支持自定义版本号 • 自动差异备份 • 支持.war/.zip • 指定更新或排除更新目录/文件
支持实时日志查看	• 代码发布进度日志 • 应用的实时日志

代码发布系统 —— 业务逻辑梳理



关于Web框架选择

Django

- 包括了几乎所有web开发用到的模块。session管理、CSRF防伪造请求、Form表单处理、ORM数据库对象化、自己的template language
- 强大的URL路由配置
- 自带admin管理界面
- 性能：★

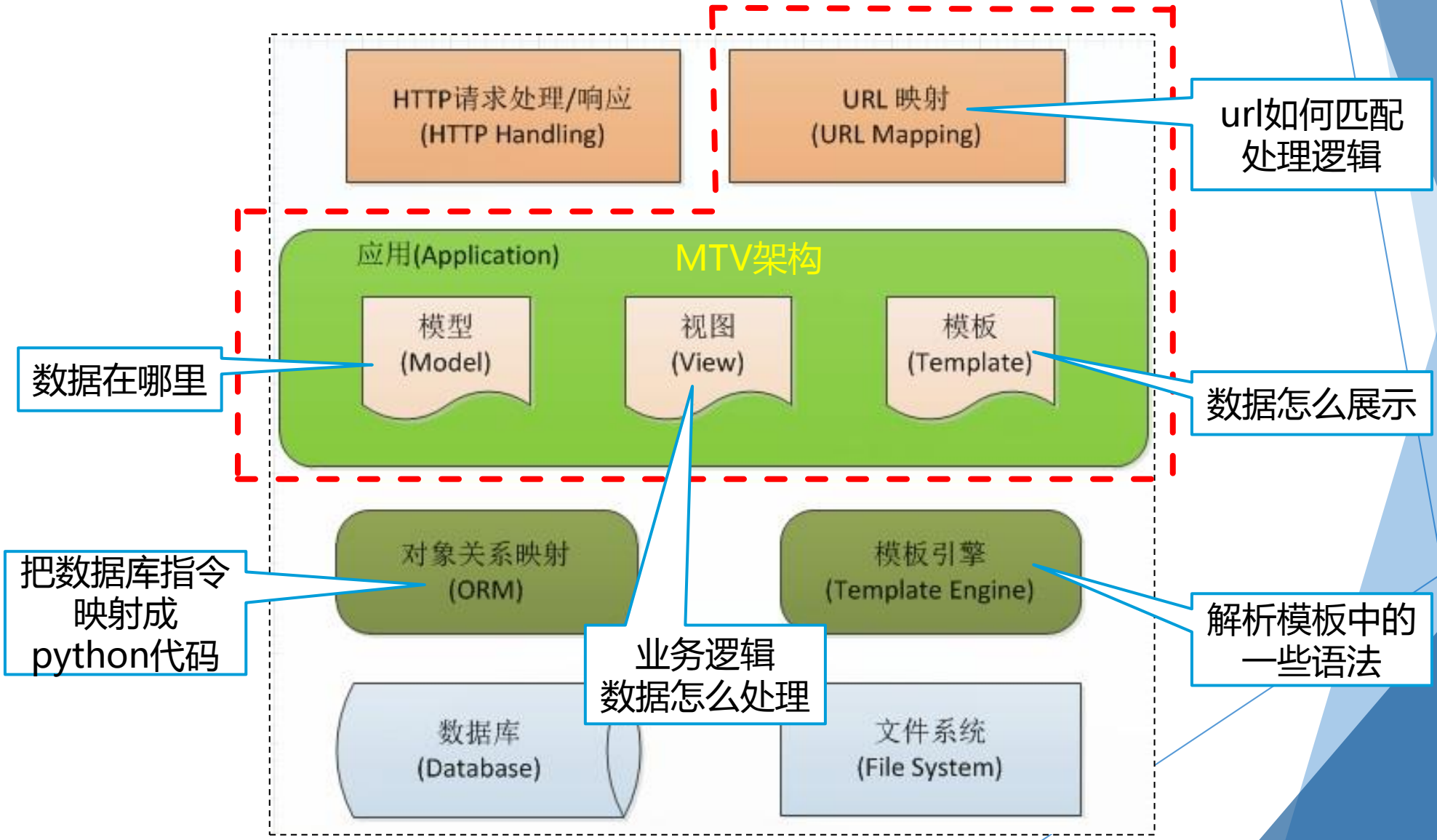
Tornado

- 拥有异步非阻塞IO的处理方式
- 有较为出色的抗负载能力、并发处理能力
- 性能：★★★

Flask

- 轻量级， 可以通过 Flask-extension 灵活扩展所需模块
- 性能：★★★☆

Django框架架构图



Django实例

用户

用户访问:
http://x.domain.com/**latest/**

模板
xxx.html

latest_books.html

```
<html>
  <head><title>Books</title></head>
  <body>
    <h1>Books</h1>
    <table>
    {% for book in book_list %}
    <tr>
      <td>书名{{ book.name }}</td>
      <td>日期{{ book.pub_date }}</td>
    </tr>
    {% endfor %}
    </table>
  </body>
</html>
```

url路由
urls.py

```
url(r'^latest/$',  'myapp.views.latest_books'),
```

视图函数
views.py

```
from django.shortcuts import render_to_response
from myapp.models import Book

def  latest_books(request):
    book_list = Book.objects.order_by('-pub_date')[:3]
    return render_to_response('latest_books.html', {'book_list': book_list})
```

数据模型
models.py

```
from django.db import models

class Book(models.Model):
    name = models.CharField(max_length=50)
    pub_date = models.DateField()
```

Django实例2 —— 加入表单

用户访问:
http://x.domain.com/add/

用户

模板
xxx.html

```
<html>
   <head><title>add_Books</title></head>
   <body>
      <h1>add_Books</h1>
      <form enctype="multipart/form-data"  action="" method="post">
         {{ form }}
         <input type="submit" value="提交">
      </form>
   </body>
</html>
```

url路由
urls.py

```
url(r'^add/$',  'myapp.views.addBook'),
```

视图函数
views.py

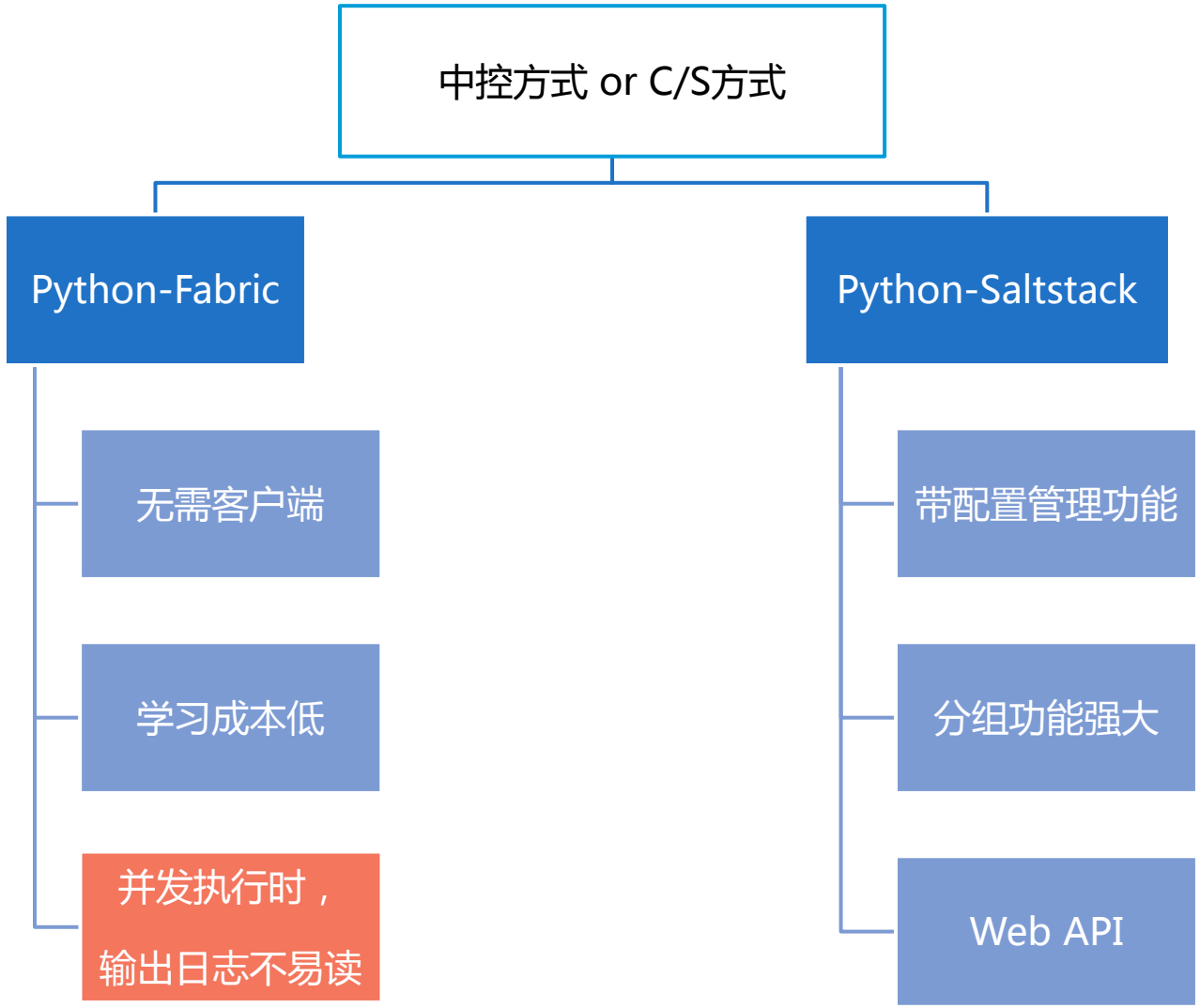
```
from django.http import HttpResponseRedirect
from myapp.forms import BookForm
from myapp.models import Book
def addBook(request):
    form = BookForm(request.POST or None)
    if form.is_valid():
            b = Book.objects.create(
                name = form.cleaned_data['name'],
                pub_date = form.cleaned_data['pub_date']
            )
            return HttpResponseRedirect('/latest/')
    return render_to_response('add_book.html', {'form': form})
```

表单定义
forms.py

```
from django.forms import ModelForm
from myapp.models import Book
class BookForm(ModelForm):
    class Meta:
        model = Book
```

数据模型
models.py

关于批量操作



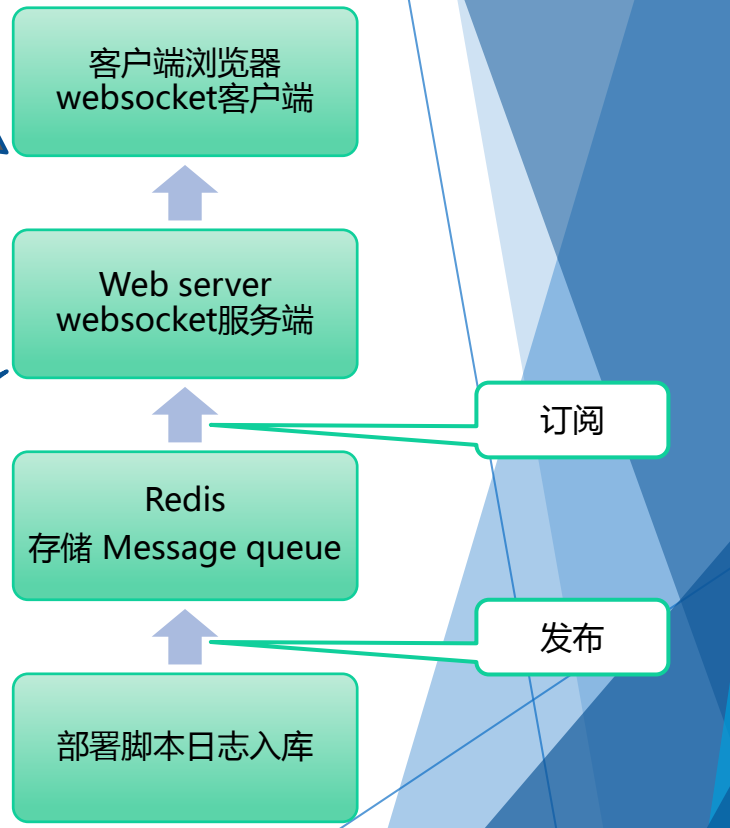
关于实时日志的展示

```
function connect() {
    ws = new WebSocket("ws://root.173ops.com/ws/deploy/{{ cfg.app.appid }}
    ws.onopen = function() {
        output('开始接收日志');
    };
    ws.onerror = function(e) {
        output('error:' + e);
    };
    ws.onmessage = function(e) {
        output(e.data);
    };
    ws.onclose = function() {
        output('服务端主动断开连接');
        ws = null;
    };
}
```

部署脚本日志

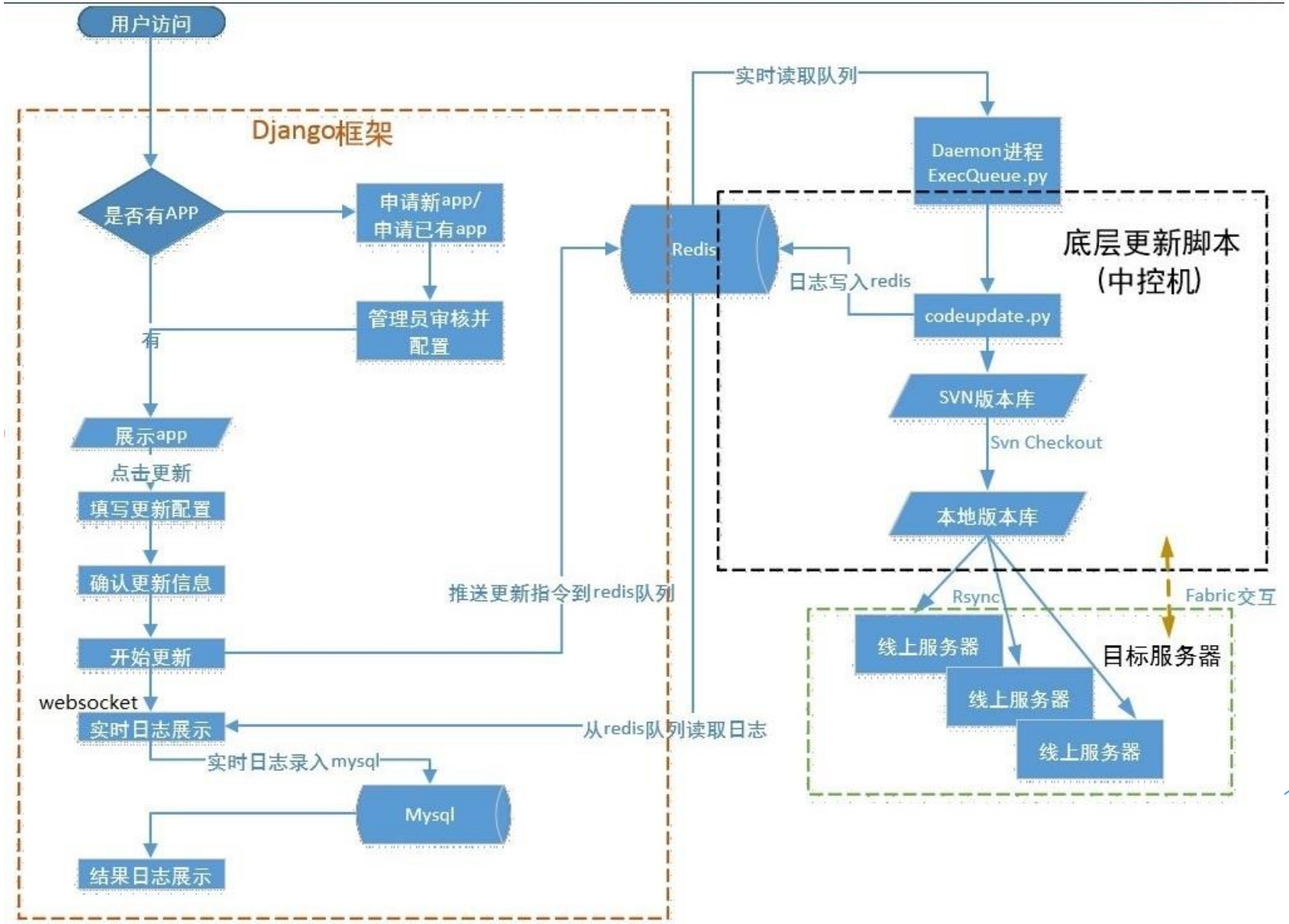
javascript脚本定期请

```
local red = redis:new()
local ok, err = red:connect("10.55.55.5", 6379)
local auth = "xxxxxx"
red:auth(auth)

local res, err = red:subscribe(channel)
while true do
    res, err = red:read_reply()
    local text = res[3]
    wb:send_text(text)
end
```




```
if record.levelno > 10:
    #只发布INFO及以上级别的日志
    self.r_server.publish(record.name, self.format(record))
```

代码发布系统 —— 架构图



代码发布系统 —— demo版

代码部署 DNS部署 项目管理 登出

选择项目 游戏数据库-cha.17173.com

代码从svn上检出后发布，代码发布系统不会做任何修改，请确认SVN上程序的数据库连接配置

代码svn地址： 对应代码根路径，目录必须以"/"结束：	http://10.5.17.181/svn/product_game_tools/code/chacha/
SVN版本号： 留空表示更新到最新版本：	
文件/目录更新列表： 用于更新特定子目录或文件 必须包含代码svn路径：	
App ID:	
目标ip:	
代码根目录:	
排除列表:	
代码用户:	
运行用户:	
需要写权限的目录:	
重启脚本:	/etc/init.d/php-fpm restart
更新策略:	只更新程序

开始更新

提升用户体验

Bootstrap模板的引入

- 与Django的模板结合，获得高大上的界面

学习一点jQuery的知识

- 选择器
- 事件触发
- Ajax请求

寻找各种插件丰富细节

- 让你的表格高大上 —— dataTables
- 让你的选择框高大上 —— Bootstrap-select
- 让你的警告框高大上 —— Messenger
- 让你的报表高大上 —— echarts
- .....

产品化所需的改进

解耦

- 页面和部署脚本解耦
- 项目和app解耦

提供API

- 单纯获取数据的操作，都通过API完成
- 统一数据返回格式，比如json
- RESTful 风格 —— Django REST framework
- 看下别人的API怎么设计

继续提升用户体验

- 产品体验超出用户期望才能形成口碑,才能传播
- 方便, 傻瓜化操作
- 高可用方案
- 提供使用文档

关于架构的改进

- 环境部署?
- 目标IP自动获取?
- 代码目录统一?
- 用户自行审核权限?
- 自动回滚?

总结 & 经验分享

- 官方手册 + Google + stackoverflow 至少解决80%的坑
- 架构清晰，科学建模，磨刀不误砍柴工
- 山寨 + 拿来主义 + 不要一步到位 == “敏捷”开发
- 选择适合自己的，就是最好的
- 多站在相关用户的角度考虑问题