Forging ahead,
scaling the BBC into Web/2.0

Dirk-Willem van Gulik
Chief Technical Architect

BBC Future Media & Technology

QCon, London, 2009

Overview

- The BBC, background and scale
- Existing path to the audience
- Forge: Change Programme, Architecture and Platform
- Summary
- Questions and Answers

ISO Stack - key to understanding Scalability

- L7: Application Layer (http, ftp)
- L6: Presentation Layer (tls)
- L5: Session Layer (sockets)
- L4: Transport Layer (tcp)
- L3: Network layer (ip)
- L2: Link Layer (ethernet)
- L1: Physical Layer (fiber, copper)

ISO Stack - key to understanding Scalability

- L9: Goals and Objectives (politics)
- L8: Organisational (process, finance, plans)
- L7: Application Layer (http, ftp)
- L6: Presentation Layer (tls)
- L5: Session Layer (sockets)
- L4: Transport Layer (tcp)
- L3: Network layer (ip)
- L2: Link Layer (ethernet)
- L1: Physical Layer (fiber, copper)

ISO Stack – key to understanding Scalability

L9: Goals and Objectives (politics)

L8: Organisational (process, finance, plans)

Once you are ‘our’ size - a lot of scaling is ‘linear’

- Bandwidth, Storage, CPU for a dynamic page
- Pure logistics - N x people/traffic/hours-on site
  - N x the ‘burden’ until you re-engineer

Organisational, Architectioal and Operational
complexity is far not linear
human comms bandwidth not easily scaled

BBC – some key facts

- 8 TV Channels (6 Digital)
- 11 Radio Networks (5 Digital)
- Funded by the license fee – £3.5bn/yr ($6bn/yr)
- 28,000 staff
- Multiple distribution platforms, Analogue, DSAT, DTT, DAB, Internet, Mobile ….
- World Service, 70 years old, 150 million listeners, 43 languages
- 233m people used BBC's global news services on tv, radio, online
- 93% of adults in the UK use BBC services
- 33m people globally use bbc.co.uk every week

Award winning, globally distributed output

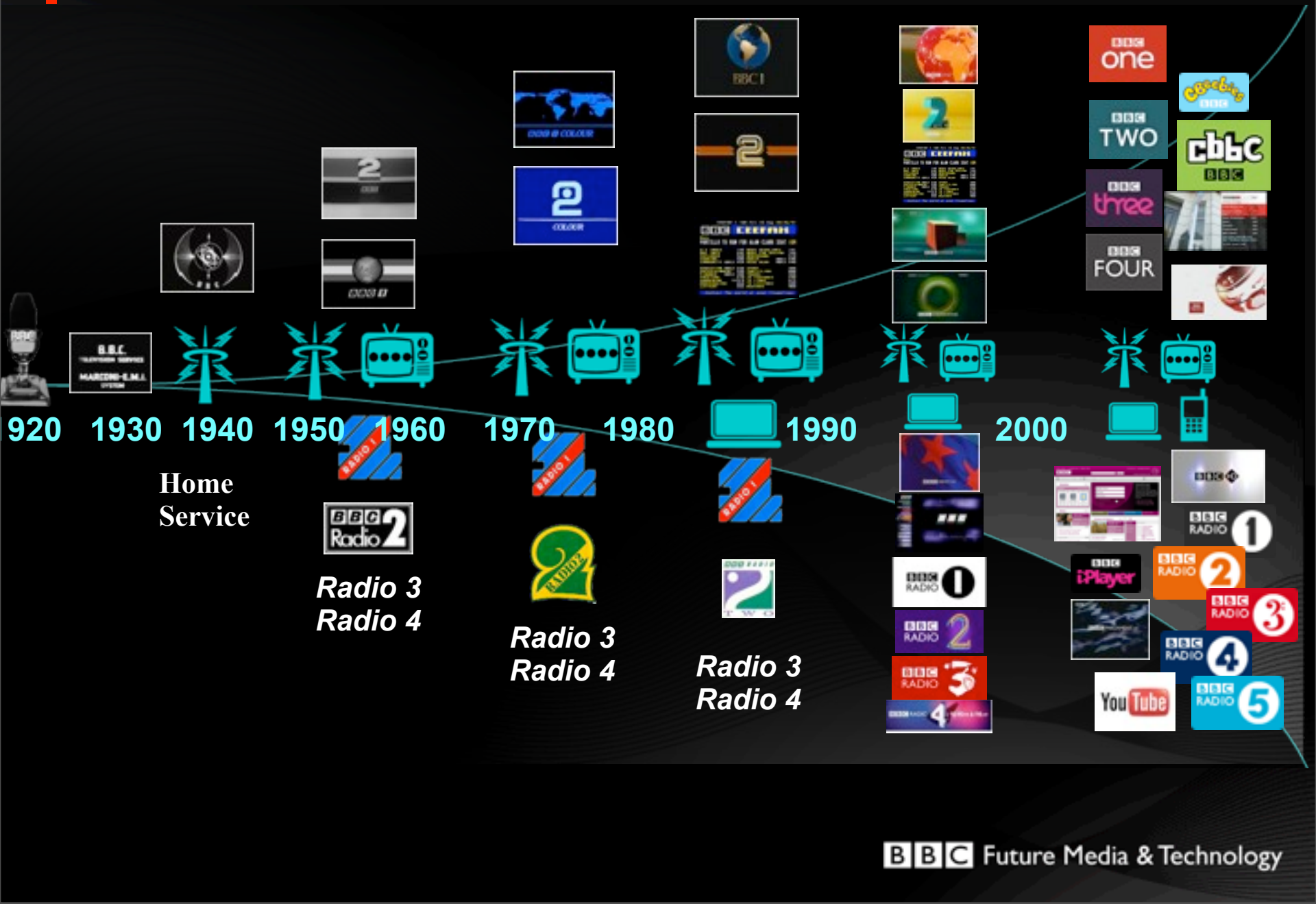
BBC Future Media & Technology

Award winning, globally distributed output

.....and **this** is our core business

BBC Future Media & Technology

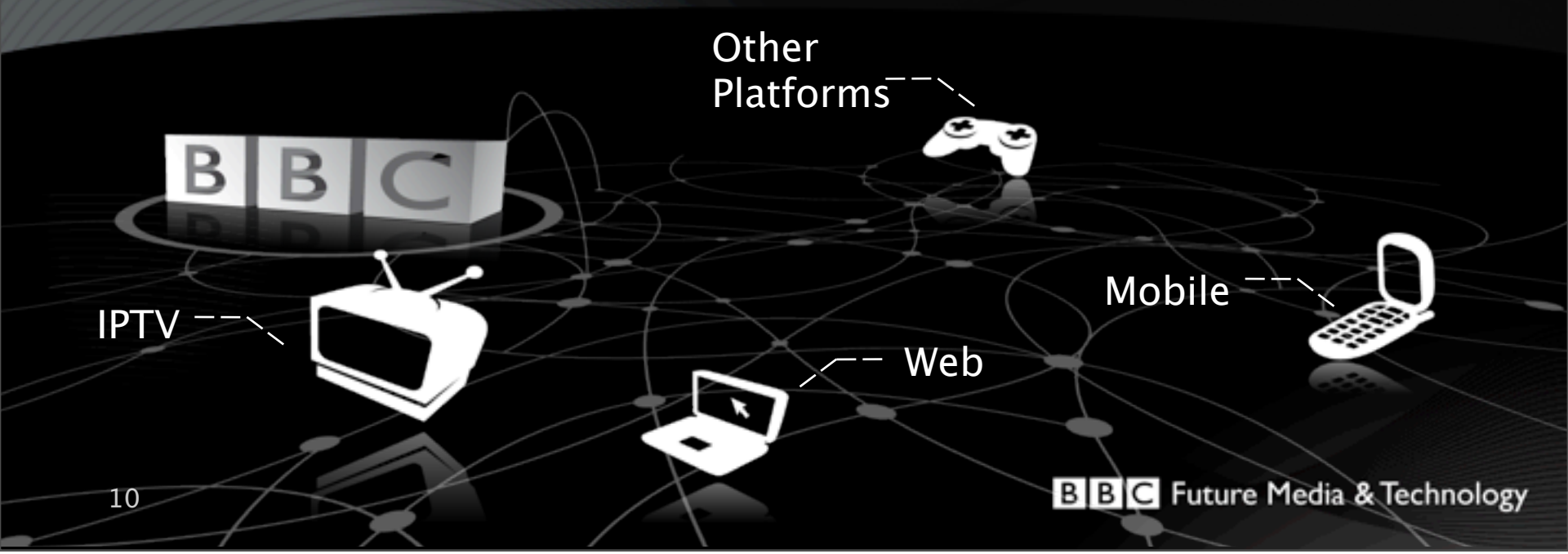
Radio to multiplatform



BBC Future Media & Technology

Rapidly evolving consumption habits

- Web, Red Button, Freeview, Mobile, iPlayer...
- Near a quarter of the IP goes to a 'TV'



Interlude: BBC Planning for growth

More users > **More** Revenue > **More** Toys?

Interlude: BBC Planning for growth

More users > ~~**More** Revenue~~ > ~~**More**~~ Toys?

The License Fee

- The BBC is funded by the License Fee

so – if:

Your site becomes **twice** as popular

then:

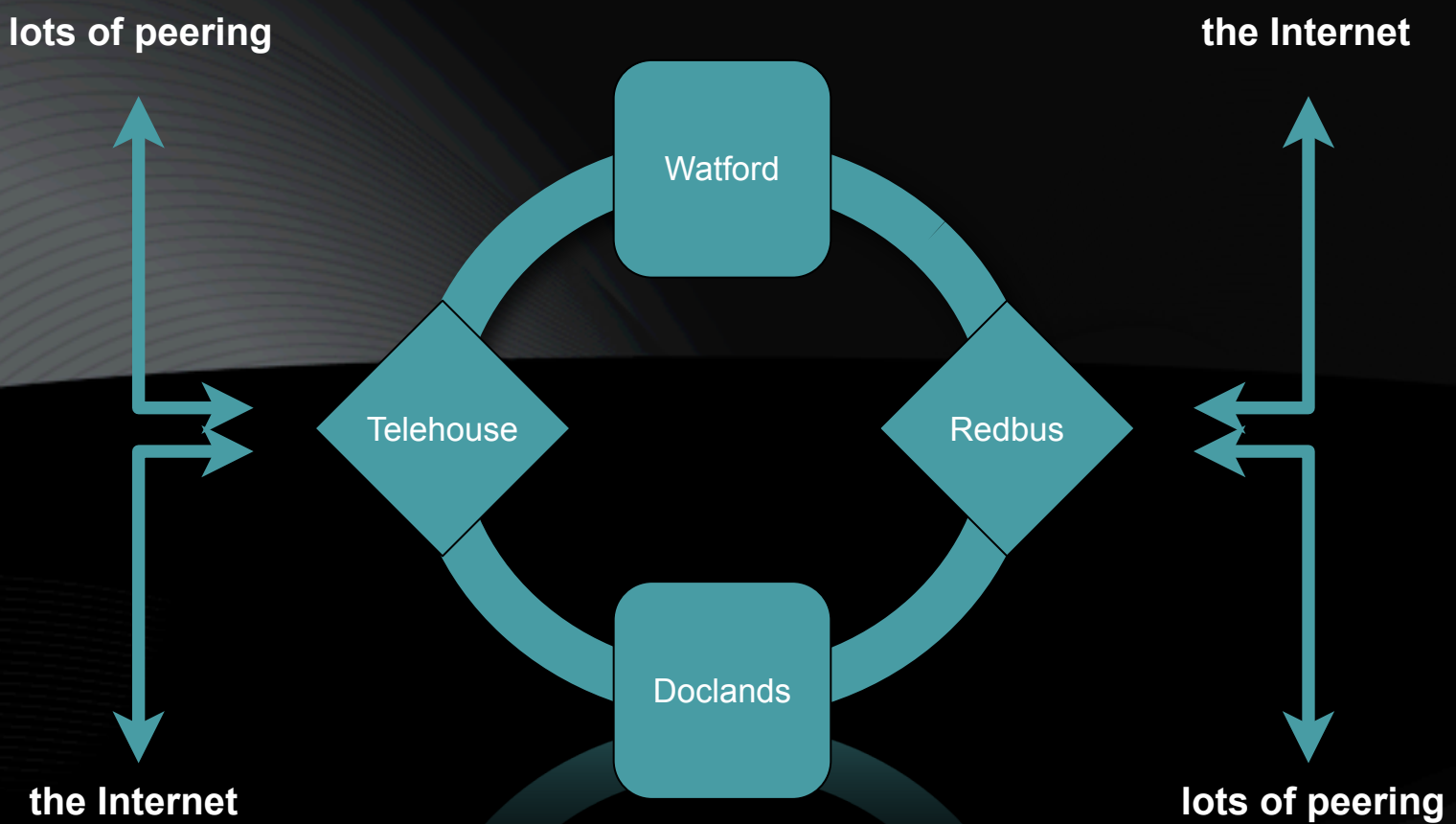
You can only spend '**half**' as much **per user** to stay within your budget.

Or in other words – we're the opposite of most commercial sites – yet are surrounded by a technical ecosystem which assumes more users is more spend.

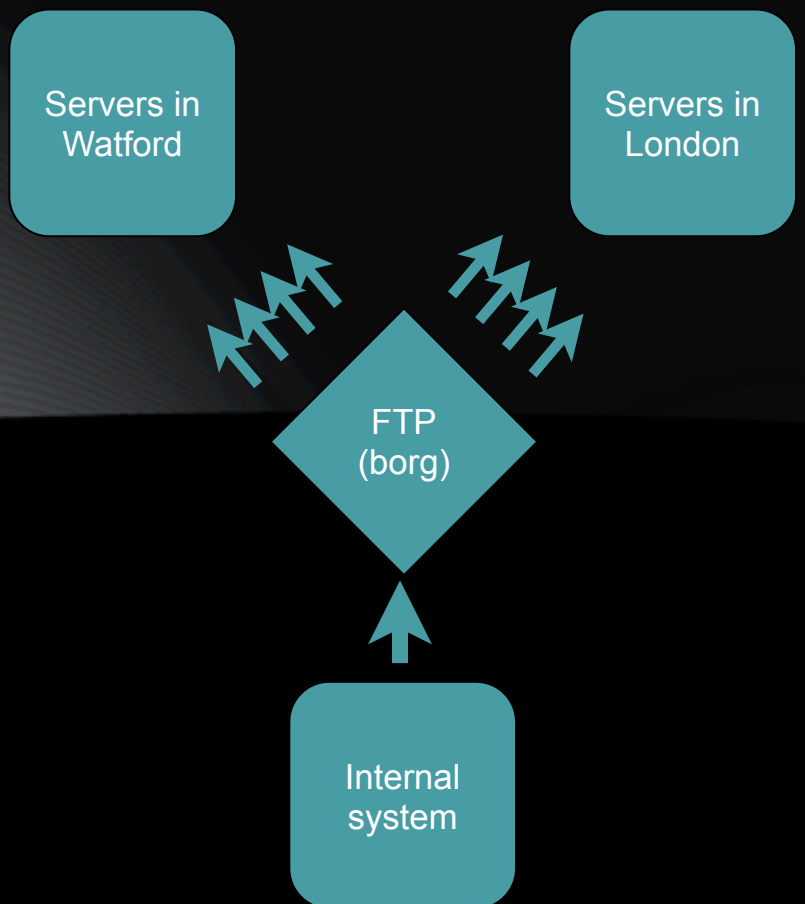
Feeding BBC.co.uk today (1/3)

- Load balanced, Round Robin DNS
- Two primary locations,
  - ~15 static, ~25 cgi machines, 100+ streaming/other.
- Solaris, some (perl) CGI
- (Static) content is ‘ftp’ed to the ‘borg’; which ftp it out to the servers
  - driven by very complex CMS generation systems.
- CDNs are pulled in where needed or cost effective
- 24x7 operations, redundant locations
- Internet peering in several locations

Feeding BBC.co.uk today (2/3)



Feeding BBC.co.uk today (3/3)



Recap

- This works actually incredibly well
- And is very cost effective
- Is ‘up’ when it counts

- Web/2.0, AJAX
- Dialogues with the user
- Customization, Personalisation, Social features
- New devices flooding the market

Forge

- From Static (1.0) to Dynamic (2.0)
  - Identity, Personalisation, Voting, Rating, Dynamic Images
- Updating technology from 20th to 21st century
  - Reusable scaleable services separated from presentation
  - Modern software stack
- Accelerated application deployment
  - Automated and repeatable deployments
- Common solutions – not inventing wheel every time
  - Common services built in a common way
- Common skills in development groups
  - Build a flexible workforce, better access to 3rd parties & simplify recruitment

What that means for us techies

- Release Engineering
  - shorten release cycles, weeks not months
- Open up the platfrom
  - don't care if dev's are internal BBC or external
  - Lower Barries to entry frameworks, abstractions, caching
  - hide complexity of distribution, multi site
- Create service platform
  - minimise wheel re invention

- Scaling mostly organisational - L8 problem
  - with a lot of help of tooling - providing a beaten path
  - remove friction and provide guidance with tooling

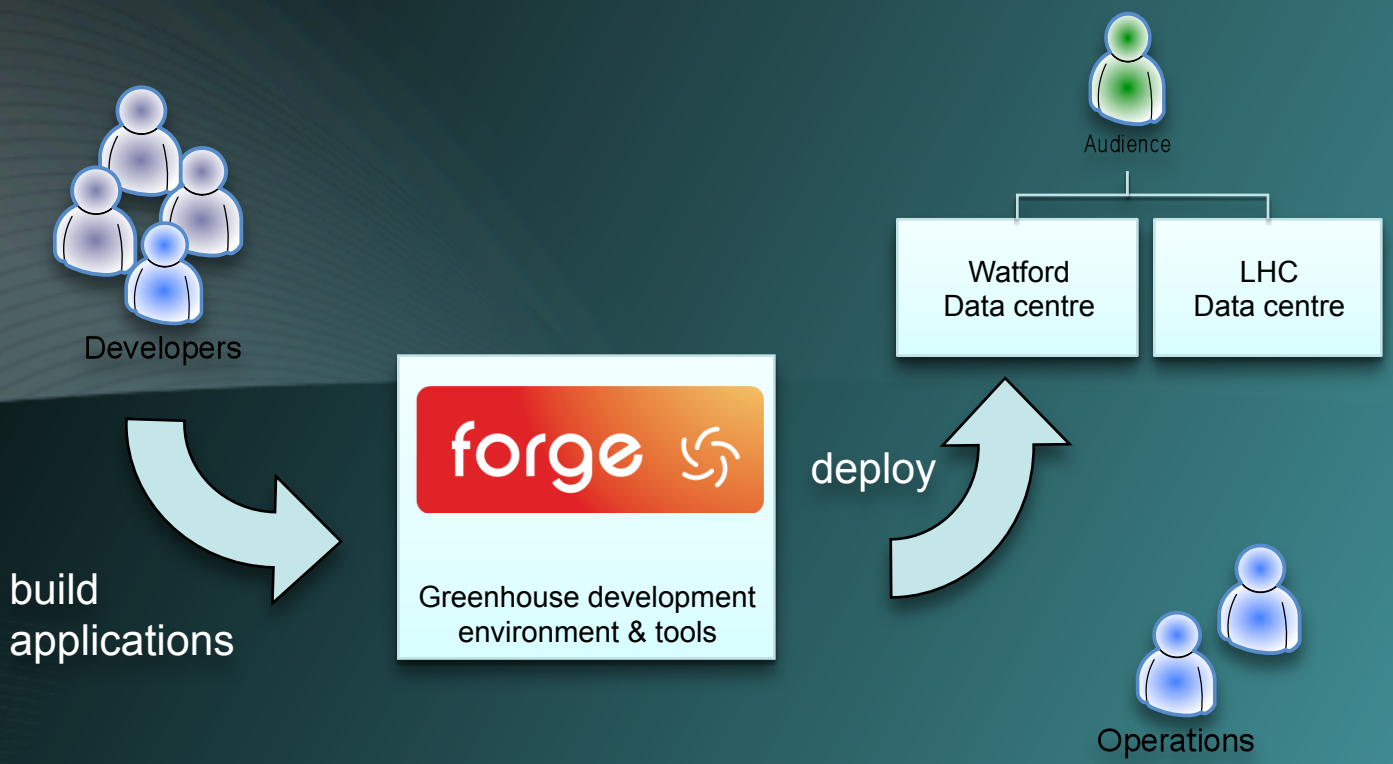
Forge: Change

- Dev tools to help ecosystem develop, improve knowledge sharing
- Transparency between dev & ops, not a wall that dev throw things over
- Dev responsible for **deployable packages** as well as code in them
- That includes a lot of release engineer

- That's the barrier to entry
  - but if you get it tight - you are on the platform in no time

...helped by a lot of tools to guide you along the golden road

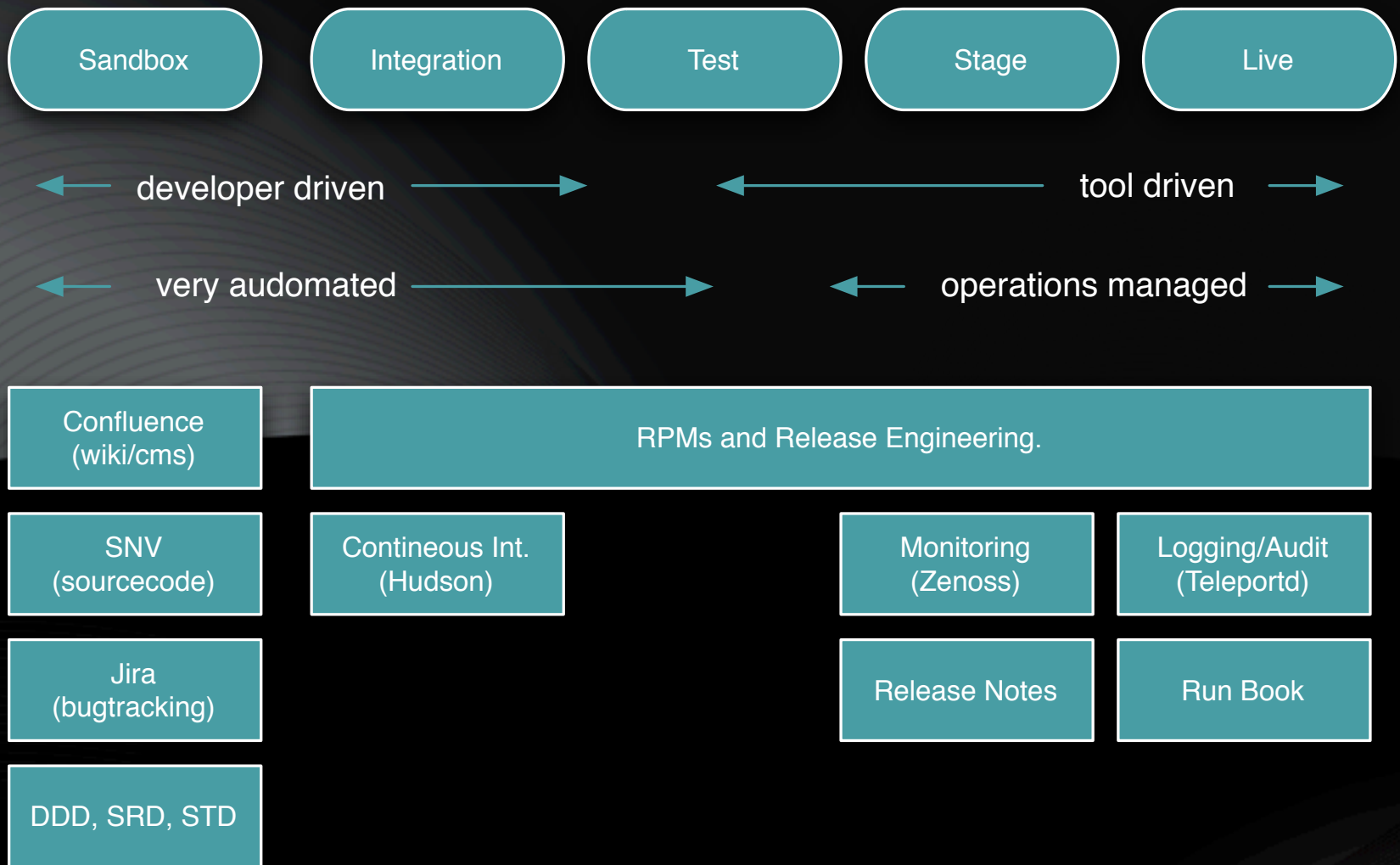
Scaling the developers



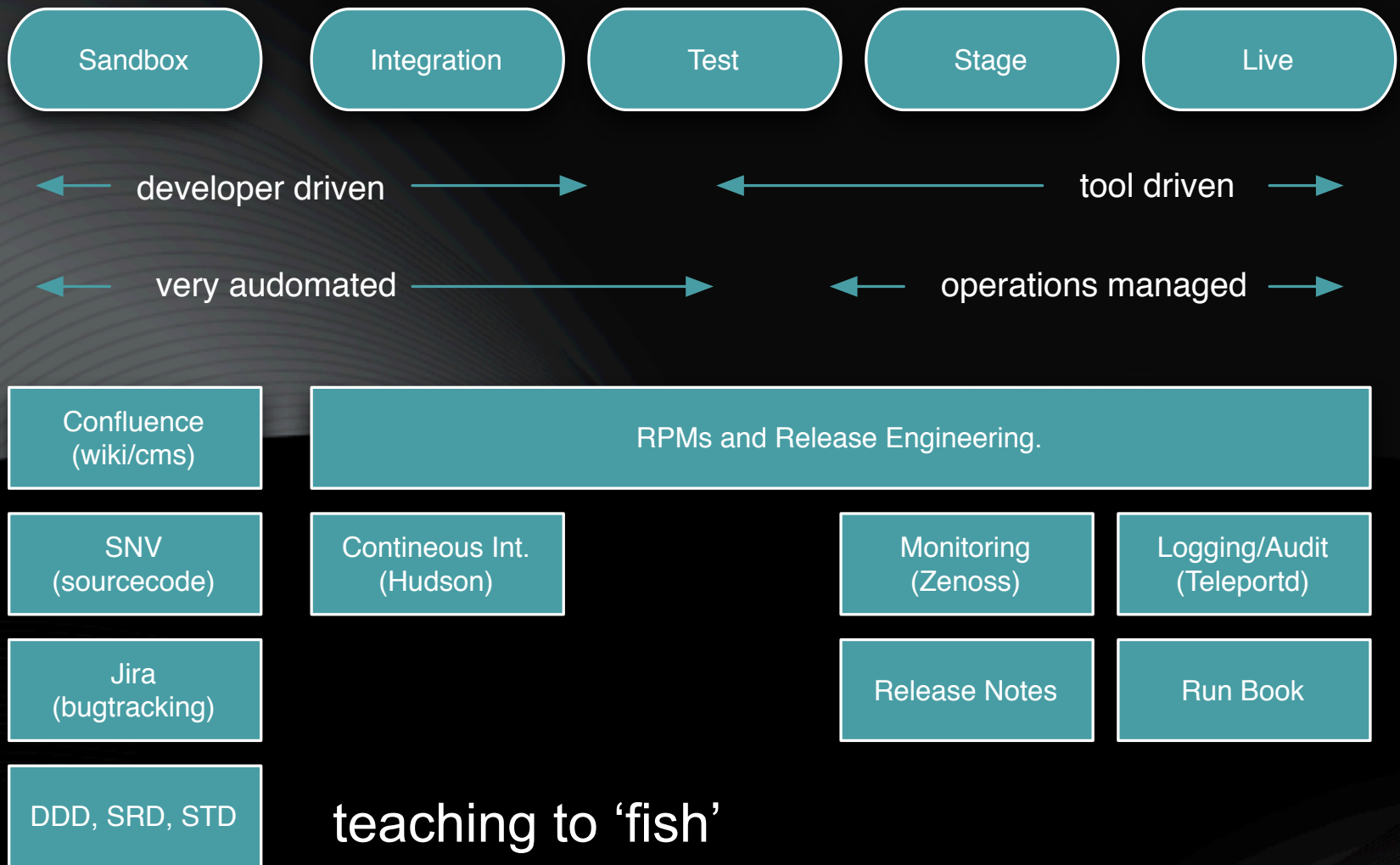
Scaling infrastructure v.s. applications

- Background and Skills of the Developers
- Complexity in the network and operations layer
- Optimize on hardware, software or complexity
  - Total Lifecycle cost can be surprising
  - 30% in development, 30% in releasing it & 30% production
  - 'Cost' of business pressure
    - Deliver extra functionality
    - Ship, Ship, Ship Now!
    - Short institutional memory
      - "Don't care about the extra ops cost"
      - "Why is this so expensive, why cannot I re-release"
- Plotting a beaten path -and automating it

From Code to Air



From Code to Air



3 Tier basic architecture

- Traffic layer
  - DNS, BGP, L7 load balancing, failover, mapping
- Presentation
  - Page Assembly Layer
  - Apache with PHP (and some memcache)
  - PHP intentionally crippled (no SQL, avoid state)
  - Optimized for 1000's of stateless requsts/second
- Services Layer
  - REST ful services for above
  - Most in Java
  - Some Perl

Layer 1 and 2

- Mostly HP C7000 chassis with blades at two sites
- Mostly Cisco lower end switches

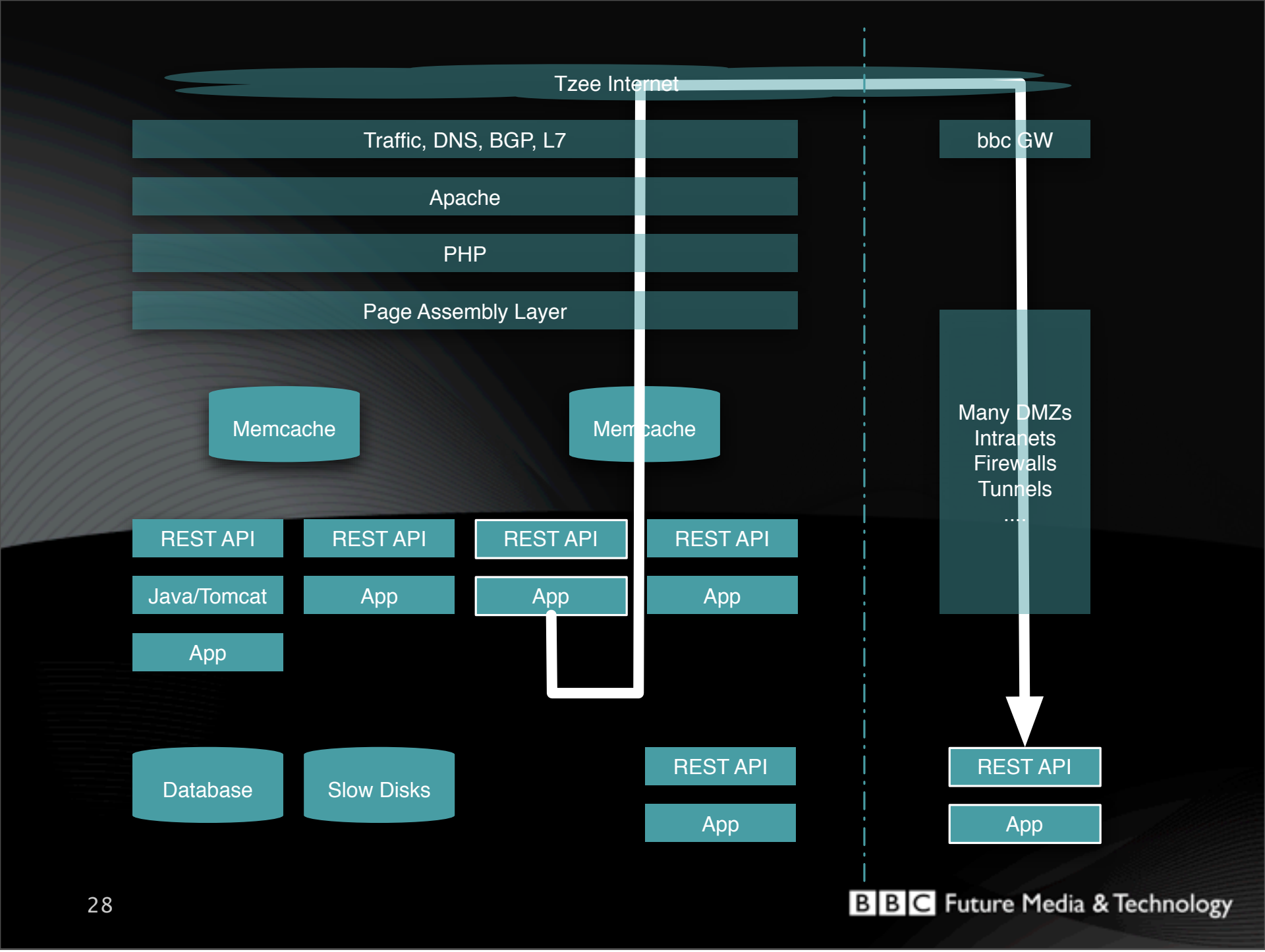
- Mostly Red Hat
- Kickstart bootstrap
- Automated, svn based

- Typical colo environment
- One and Ten Gbit
- Bonding on Aggregation
- Keep it Simple

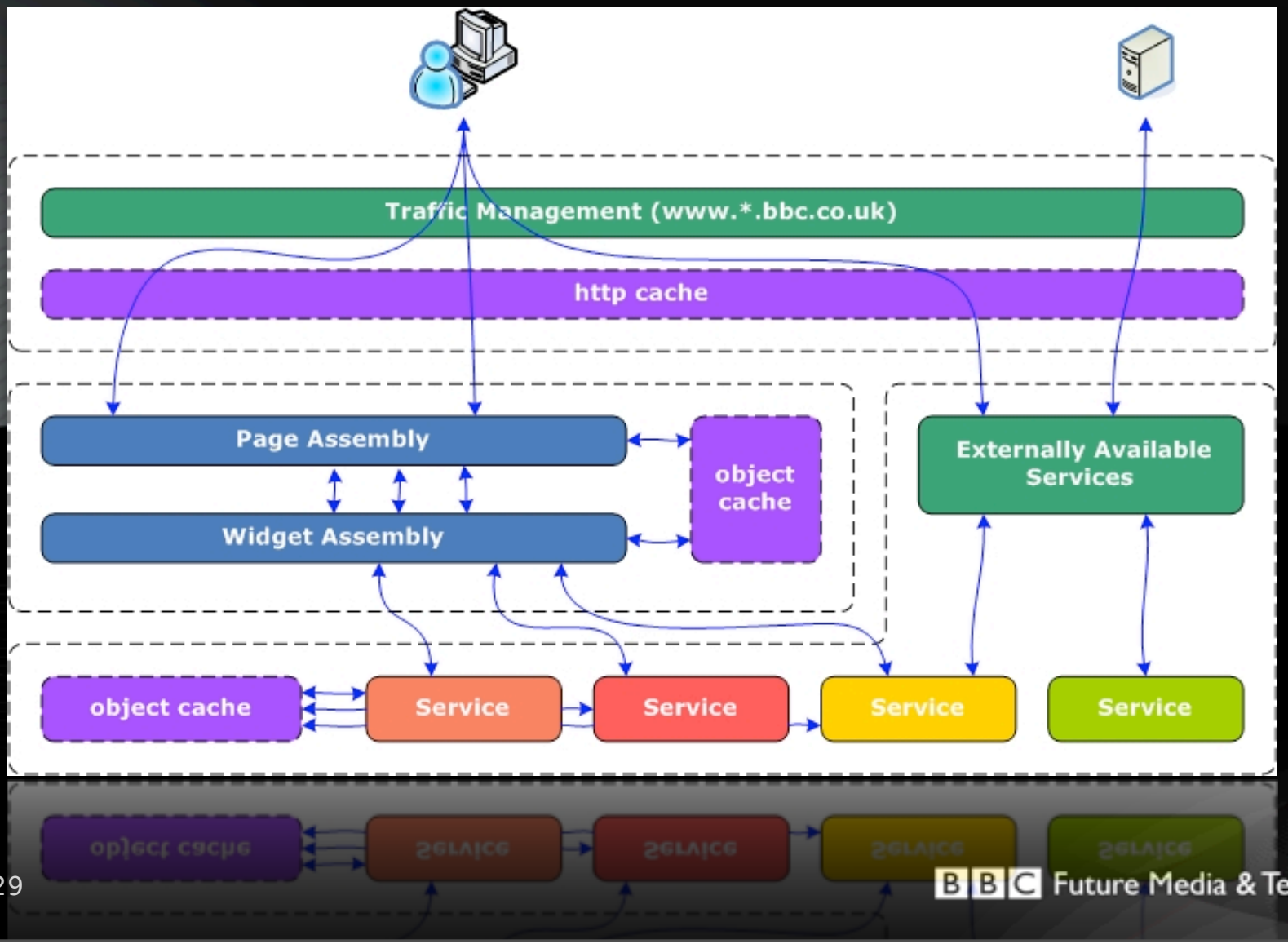
- stateless
- quick

- stateful
- few hits

- intersystem RESTfull


Tzee Internet
Traffic, DNS, BGP, L7
Apache
PHP
Page Assembly Layer
bbc GW
Memcache
Memcache
Many DMZs
Intranets
Firewalls
Tunnels
....
REST API
REST API
REST API
REST API
Java/Tomcat
App
App
App
App
Database
Slow Disks
REST API
App
REST API
App

and in more detail



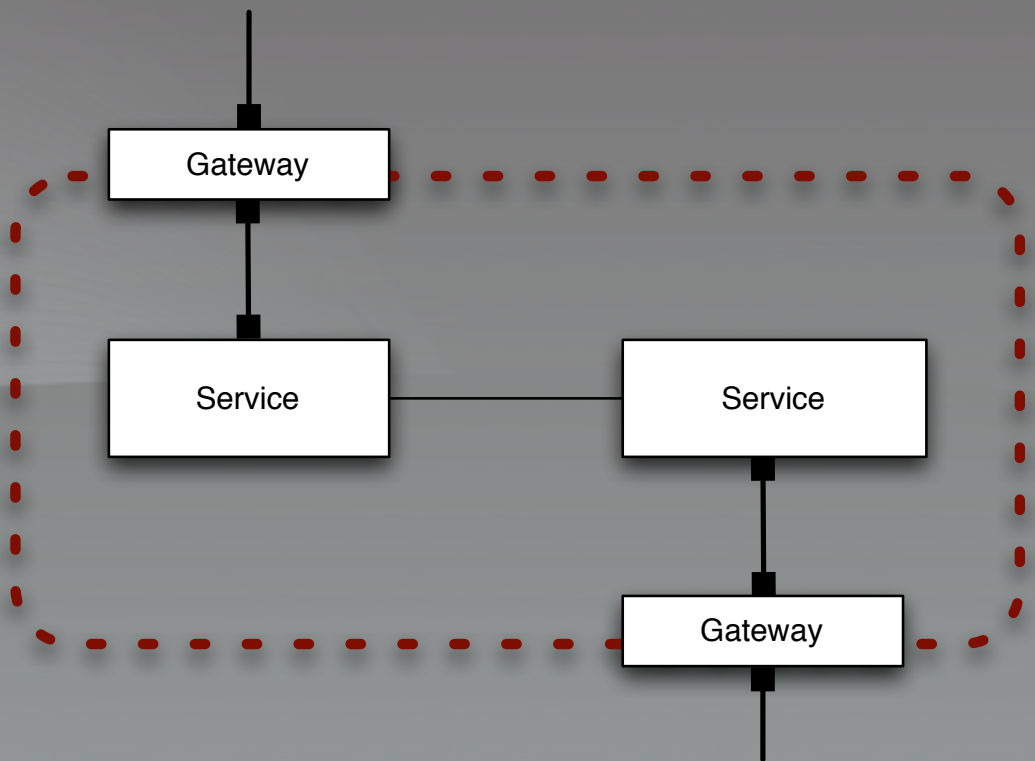
Why REST matters

- Lots of services, sites, systems, “Releases” several times a day, Applications are moved, shuffled
- ‘Allow from 192.168.1.20/24’
  - what do you allow ?
  - and what mix of applications & data
  - Significant compliance complexity
  - one bad apple can spoil the whole barrel
- **Omnipotent protocols (ssh, sql) are evil!**
- ‘Who’ does ‘What’ to ‘Which’ data
- REST – do ‘exactly’ what is says on the tin

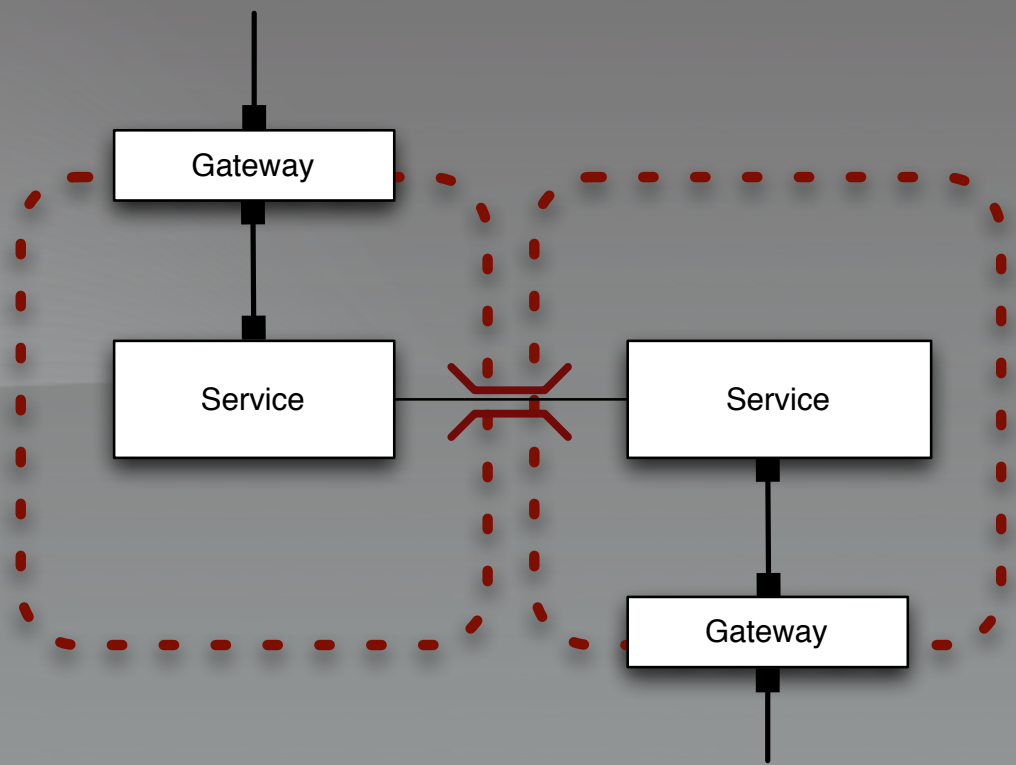
Contamination – the scaling problem

- Really a function of organisational structures
  - Policy, Governance and Compliance
- Lots of applications, lots services
  - Different departments, different timelines
- You cannot test everything against everything
- Full scale code audits are expensive
  - And no longer realistic in a SaaS world.
- Locking everything down seems cheap
  - But carries a horrible ‘agility’ cost
  - And risk of escalating ‘red tape’ spirals
- Grouping in ‘zones’ solves part of the problem
  - but one bad apple can spoil the whole barrel

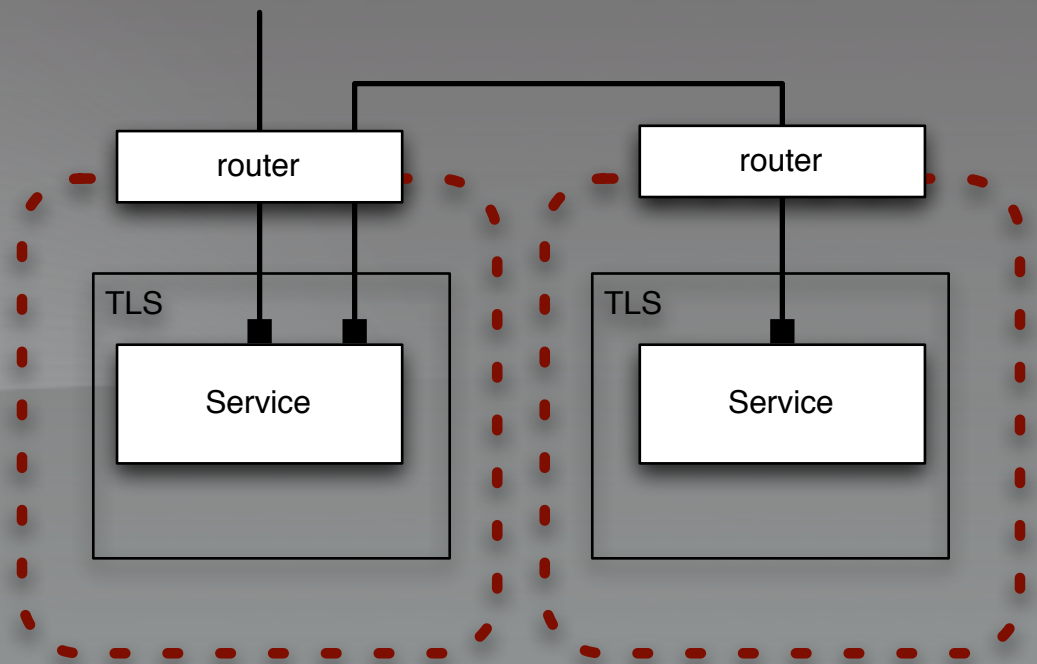
Containments (1/3)



Containments (2/3)



Containments (3/3)



Couple of 'PKI' tricks

- Public Key Infrastructure
  - 'Authorities' vouch for 'Siblings'
- Very explicit trust statement
- Validation possible without knowing a secret
  - very little (if any) data truly sensitive
  - lack of validation – failsafe choise (allow or deny) open
- Validation possible 'off-line'
  - if you accept a certain amount of slack
  - which matches with biz-choices in times of dire need.
- No need for a highly available central directory
- Solves only part of the 'AAA'
- Allows one to solve the 2nd A far cheaper.

Baseline Services

- Baseline REST services key
  - prevent re-inventing the wheel
- KV store - with abstract interfaces
  - RDBMS-es are expensive
- RDBMS - cross site sync, backup
- SVN - auto updates and free 'audit'
- Page Assembly Layer
- RPM packaging
- Tomcat packaging

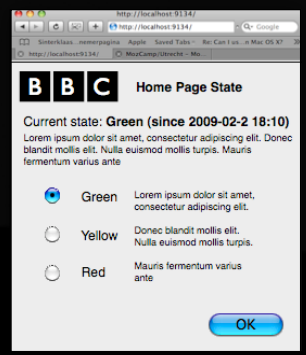
- Monitoring, Memcache, Spread, Flapole, Logging and may more...

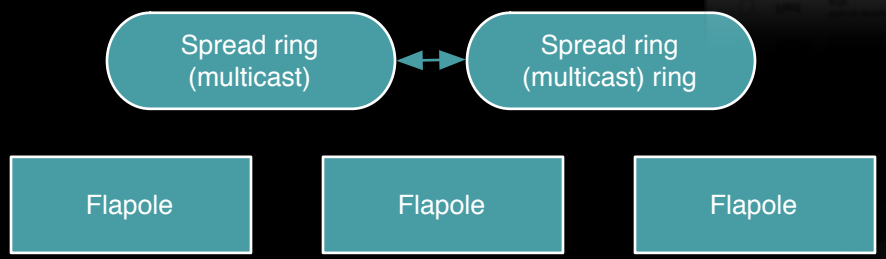
Multi Site – Split Brain Headaches

- Multiple Sites
  - One can go down ?
  - Stickyness is expensive, as are global sessions
- Is the other site ‘really’ down
  - It can’t be me – I know I am ‘up’ !
- After the disaster – halfes rejoin
  - No rejoicing – doube difficult period
  - Sync databases, move sessions,
  - Blow Caches ? Rebuild ?
  - Thundering herds
  - All at the same time
- Good upfront detailed design
  - and some help...

Fla(g)poled

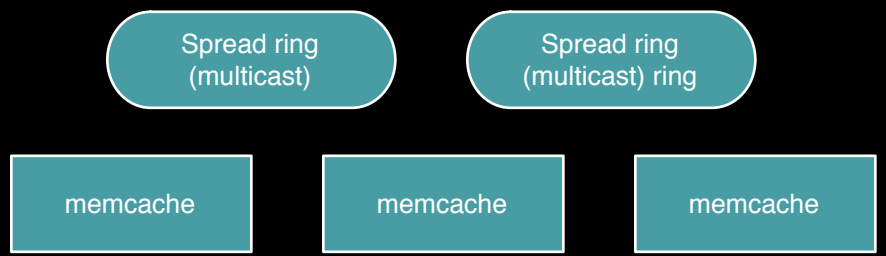
- Every machine has a flagpole; every app can see it
- Spread interconnected
- REST api, ‘touch’, ‘test -e’ API
- Application specific rules:
  - when ‘red’ – don’t personalise, drop images
  - when ‘orange’ – do not purge your cache
  - when ‘yellow – offload to CDN if possible
  - when ‘split brain’ – tolerate stale caches





Memcache and Split Brain

- Memcache – its like duct tape !
- Very low cost cache, very fast
- BUT only because it is simple and not distributed
- 'full contract' – everything is everywhere and fresh
  - very expensive in multi site
- Second best: 'never return something state'
  - Definition of never is complex during 'split brain'

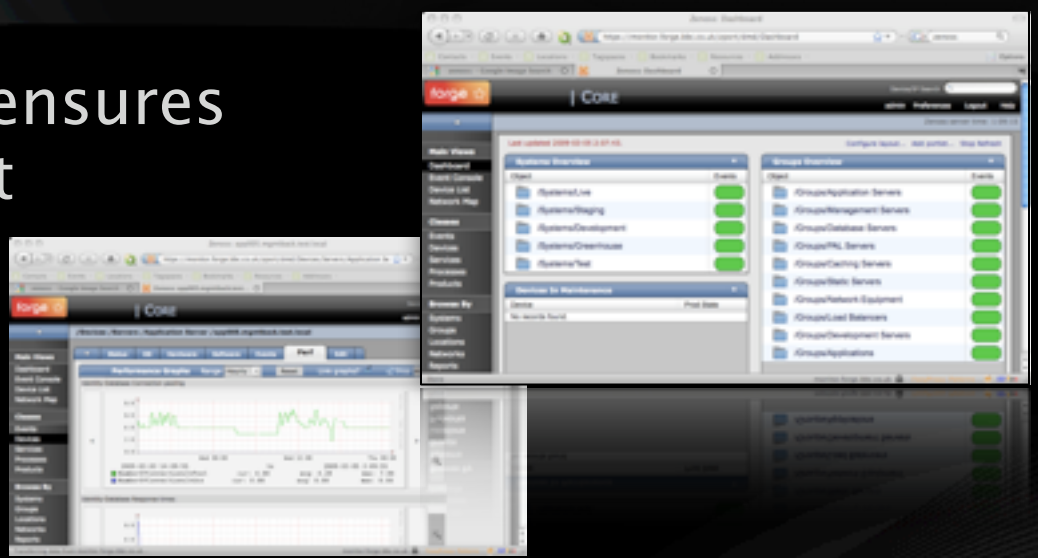


Monitoring

- SNMP (remember: no omnipotent protocols)

- Use of ‘v3’ – strong authentication/confidentiality
- Connectors to java, shell scripts
- Wrappers (e.g. for cron)

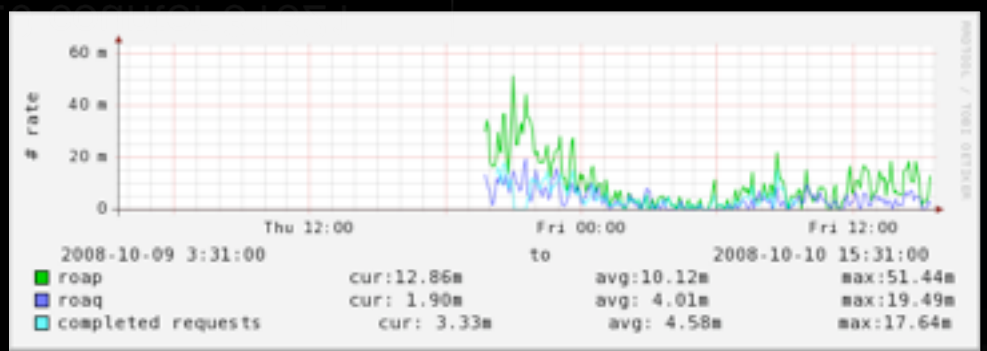
- SNMPv3 ‘boundary’ ensures that the path of least resistance ‘scales’



SNMPv3

- JMX or ‘flat file’ picked up by net-snmp
- OIDs allow for co-existence (RPM police)

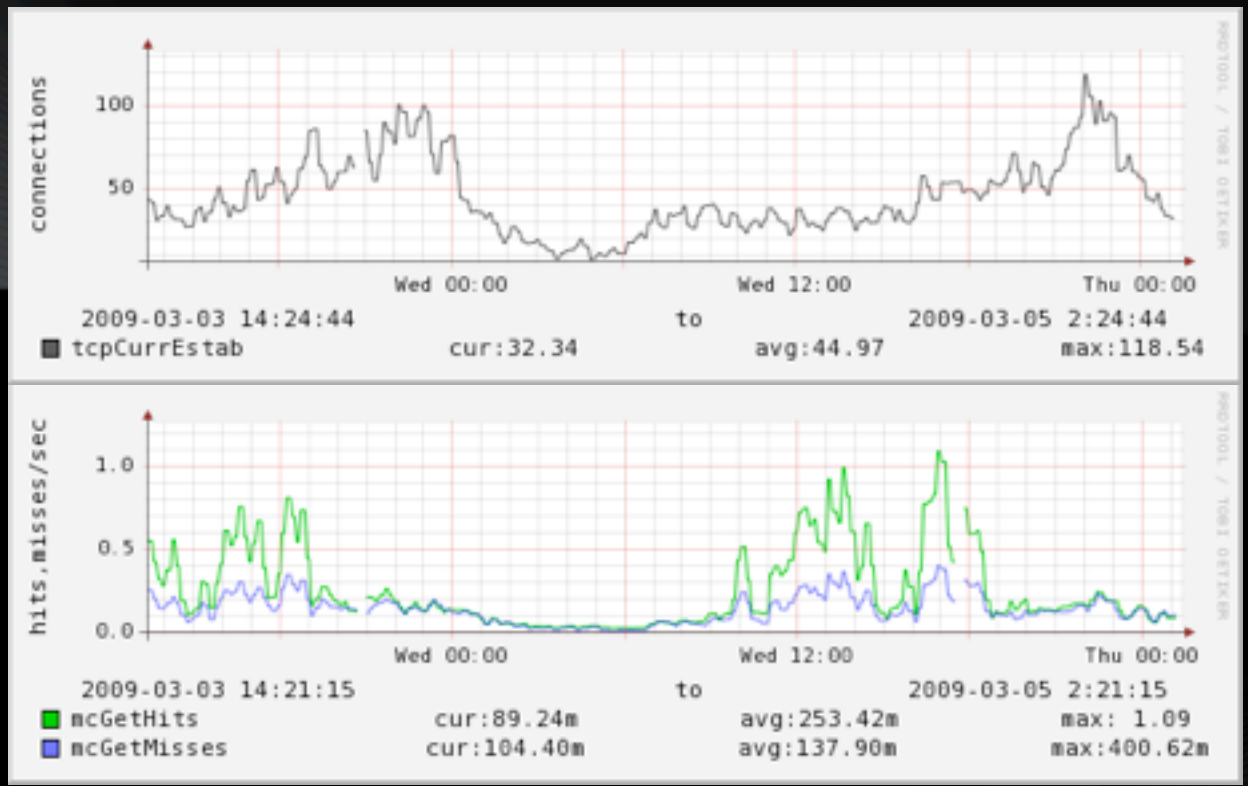
```
1.3.6.1.4.1.30754.2.1.5.2.3.1.0 counter 1001
1.3.6.1.4.1.30754.2.1.5.2.3.2.0 counter 1234
1.3.6.1.4.1.30754.2.1.5.2.3.3.0 counter 12312321
1.3.6.1.4.1.30754.2.1.5.2.3.4.0 counter 31321
1.3.6.1.4.1.30754.2.1.5.2.3.5.0 gauge 301
```



BBC Future Media & Technology

SNMP

- Allows for correlation

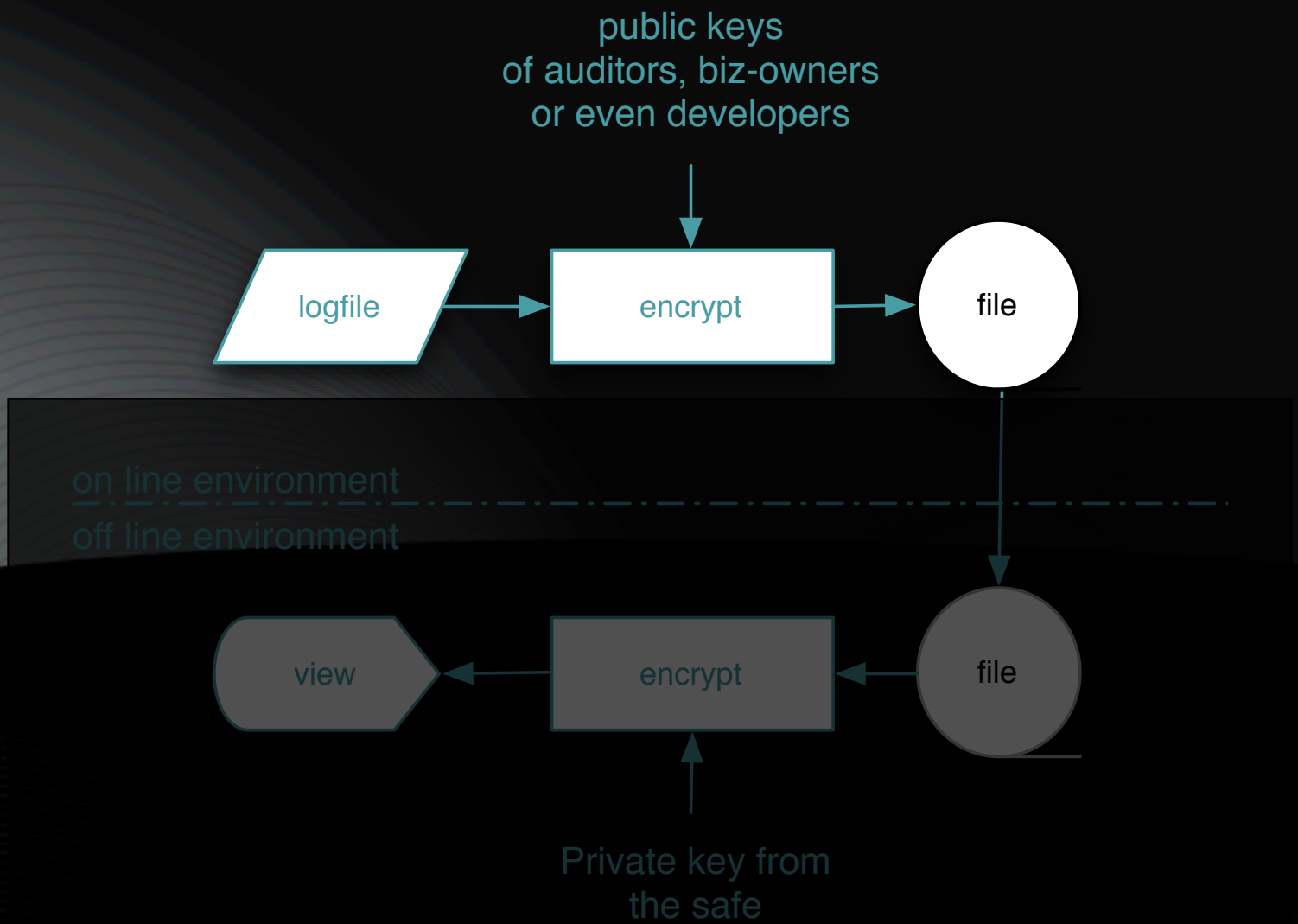


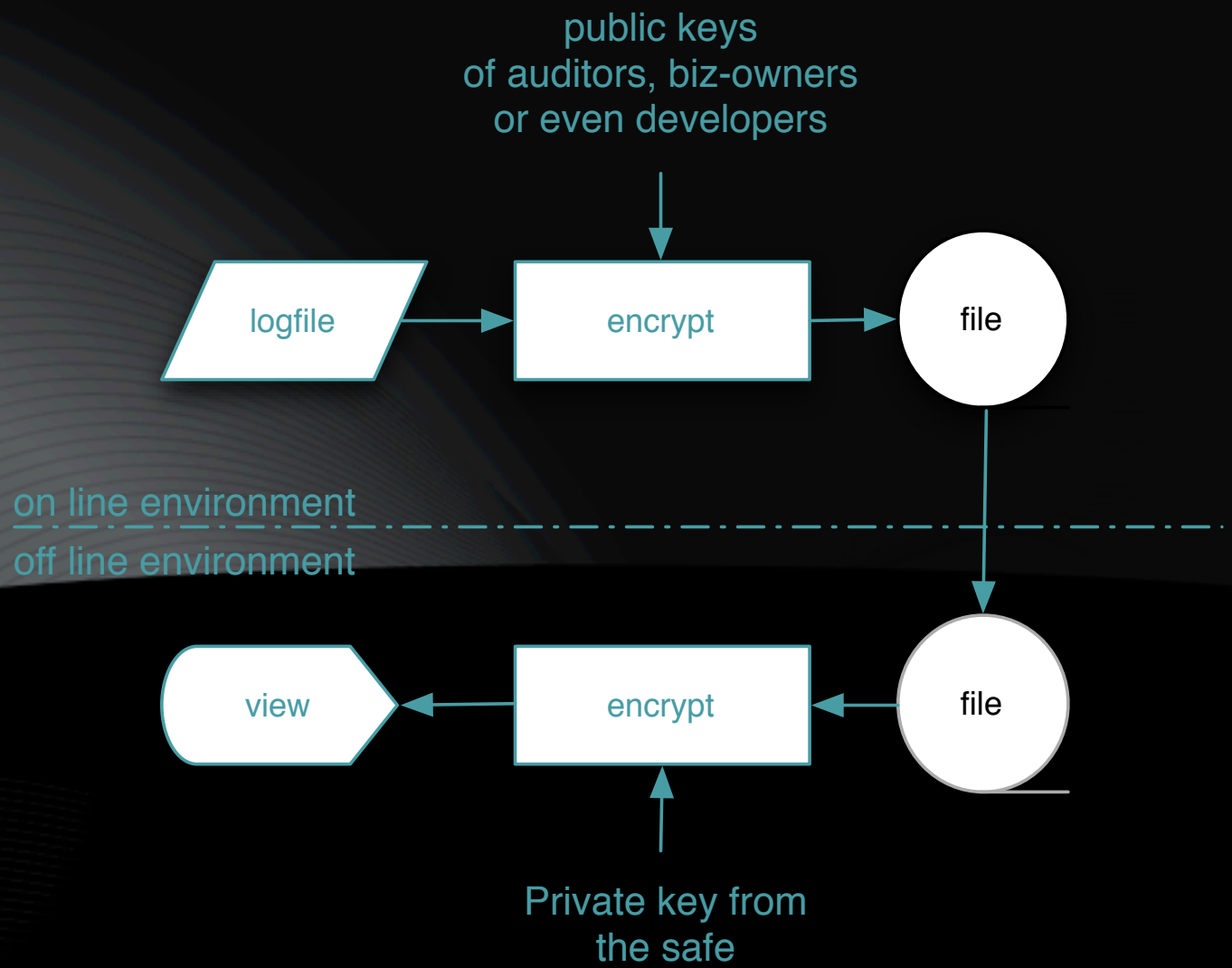
Databases & Omnipotent

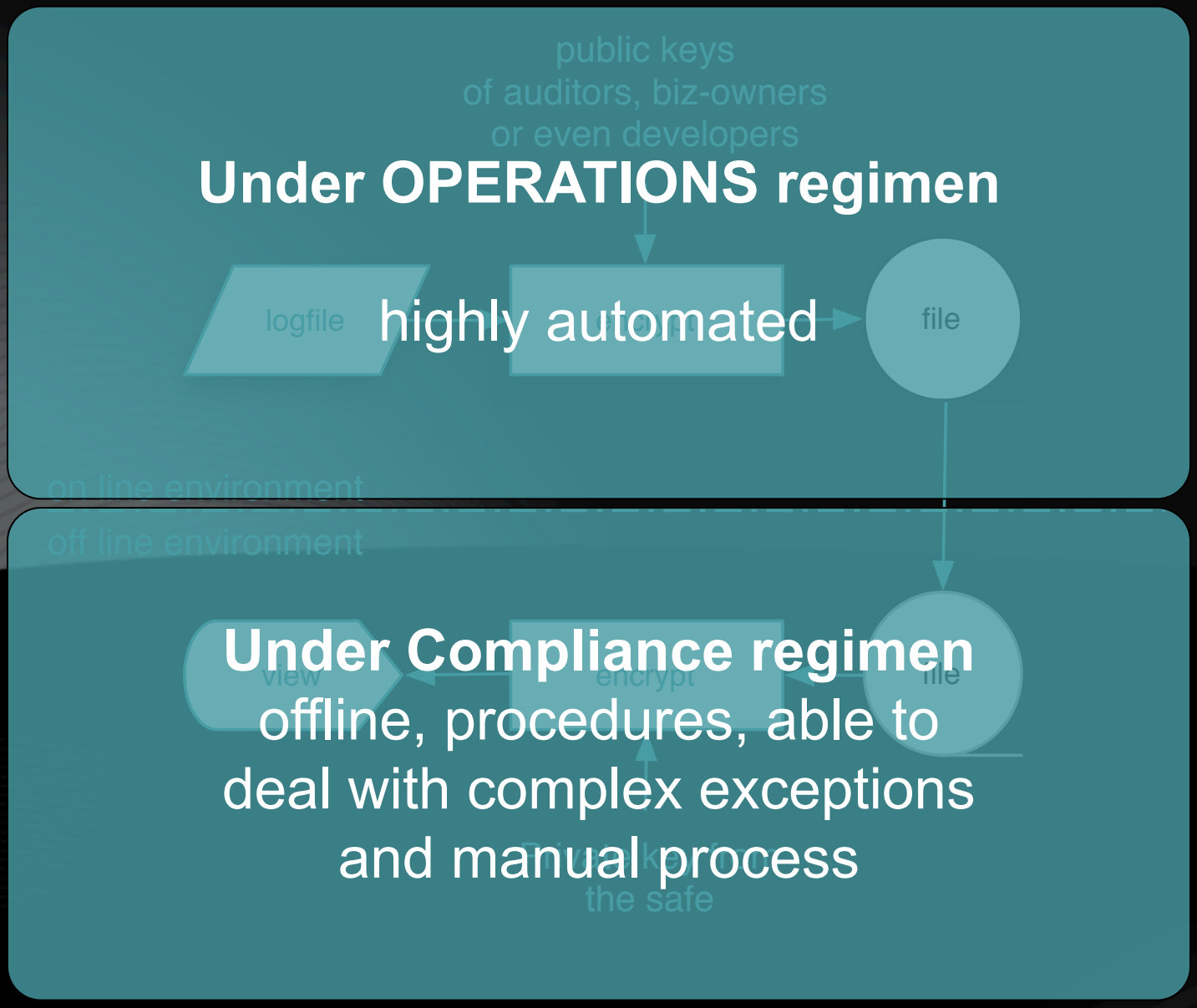
- Slides for P2–9, and P7–10

Audit logging 1/2

- Logfiles can contain sensitive data
  - Data Protection Act
  - IP addresses, timestamp, postcode
  - Forum contributions, search strings
- Layered/Zoned security model
  - even the smallest morsel from a higher class ‘wins’
- Such contamination is expensive
  - Ensuring compliance, training, limited access
- Scaling of a different sort
  - What if you did not have to worry at all ?
- Trapdoor logging


public keys
of auditors, biz-owners
or even developers
logfile
encrypt
file
on line environment
off line environment
view
encrypt
file
Private key from
the safe


public keys
of auditors, biz-owners
or even developers
logfile
encrypt
file
on line environment
off line environment
view
encrypt
file
Private key from
the safe


public keys
of auditors, biz-owners
or even developers
Under OPERATIONS regimen
logfile
highly automated
file
on line environment
off line environment
Under Compliance regimen
offline, procedures, able to
deal with complex exceptions
and manual process
view
encrypt
file
the safe

Teleporting (1/2)

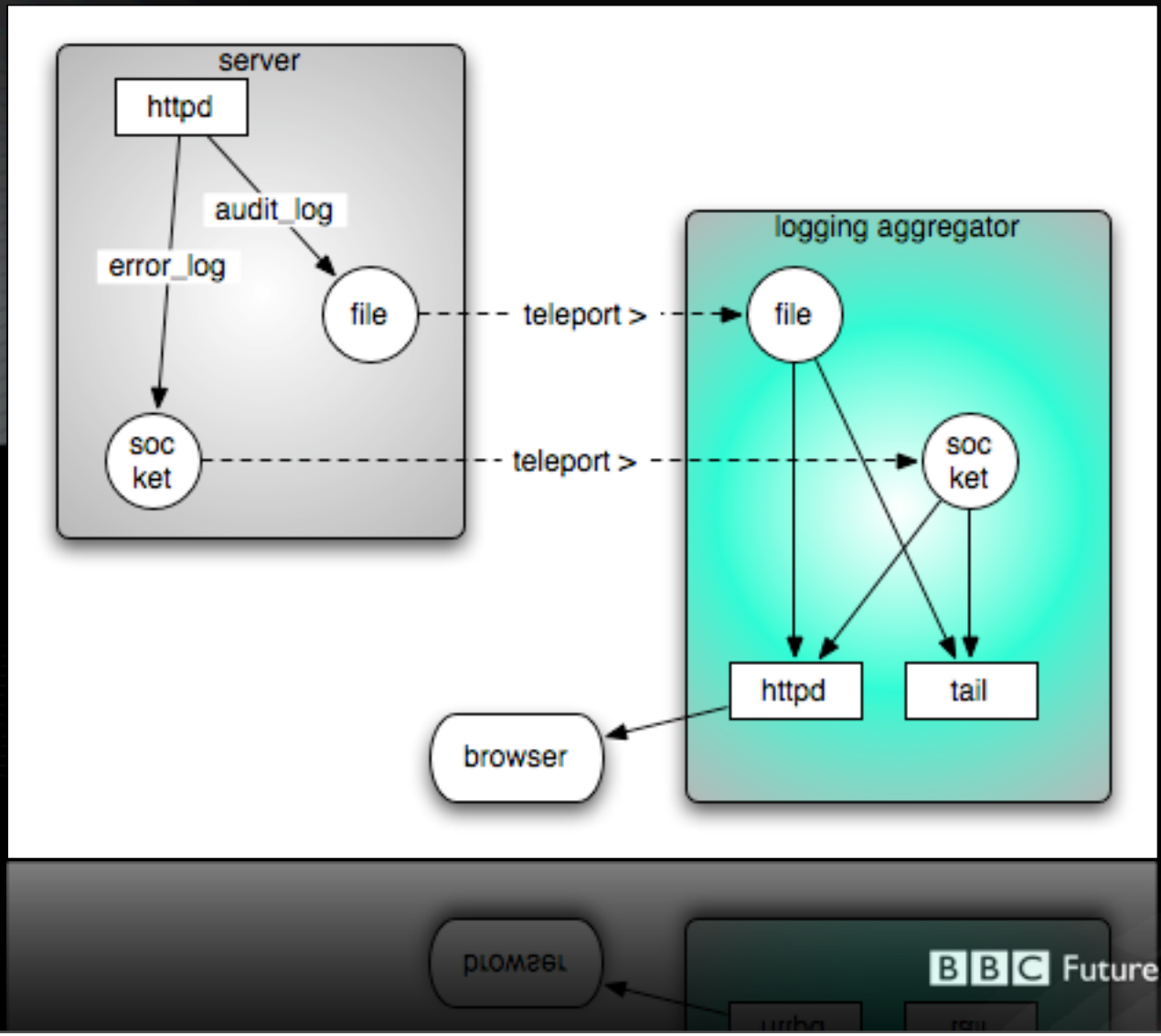
- Logging – many different functions

- Some of it very transient
- Multi site cluster
- Multiple tenants

Teleport

- ‘On demand’ map through
- As simple ‘ttail –f <log id>’
- Small (https) web portal
- AAA simple – trapdoor logging.

Teleporting (2/2)



- steering group slides last - 3

All Combined

- Front end
  - Simple PHP environment
  - All nasty things taken care off (and memcache to boot)
- Back end
  - Simple REST service
  - Monitoring, Logs, Audit, Distributed, State – 'free'
  - Substrate, App server – all taken care off
- Platform
  - SVN, Jira, Release engineering – all tightly coupled
  - Automated build, test, packaged deployment
  - Logs, Configs & State – visible to all

To Recap

- Scaling
  - If you are the size of the BBC - the real challenge is L8/9
- Tools and a friction free 'beaten path'
- Provide absolute transparancy to everyone
  - Only then can developers (and their managers) get involved and understand operational ramfications
  - Only then do you get visibility of the total-lifecycle cost
- Detailed Design absolutely crucial
- Avoiding Contamination
  - win/win for ops, devs and compliance
- Process and Release Engineering
  - much more important than cool code

Questions ?

Dirk-Willem.van.Gulik@bbc.co.uk

Tag Cloud

- Apache HTTPD
- PHP
- Tomcat
- CouchDB, Voldemort
- Spread
- Memcache
- Mysql (postgress)
- Zenoss
- Openssl
- Jira, Confuence, SVN, Hudson