High Availability Explained

Maciej Lasyk

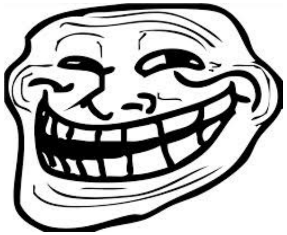
11. Sesja Linuksowa

Wrocław, 2014-04-05

“Anything that can go wrong, will go wrong”

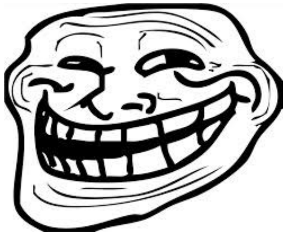
Murphy's law

“Anything that can go wrong, will go wrong”



Murphy's law

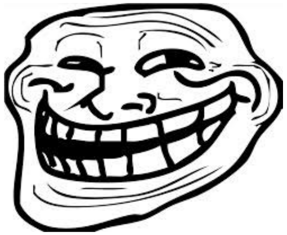
“Anything that can go wrong, will go wrong”



Murphy's law

An electrical explosion and fire Saturday at a Houston data center operated by The Planet has taken the entire facility offline. The company claimed power to the facility was interrupted when a transformer exploded. Official reports that three walls were blown down causing a fire.

“Anything that can go wrong, will go wrong”



Murphy's law

An electrical explosion and fire Saturday at a Houston data center operated by The Planet has taken the entire facility offline. The company claimed power to the facility was interrupted when a transformer exploded. Official reports that three walls were blown down causing a fire.

Three walls of the electrical equipment room on the first floor blew several feet from their original position, and the underground cabling that powers the first floor of H1 was destroyed.

High Availability is in the eye of the beholder

High Availability is in the eye of the beholder

CEO: we don't loose sales

High Availability is in the eye of the beholder

CEO: we don't loose sales

Sales: we can extend our offer basing on HA level

High Availability is in the eye of the beholder

CEO: we don't loose sales

Sales: we can extend our offer basing on HA level

Accounts managers: we don't upset our customers (that often)

High Availability is in the eye of the beholder

CEO: we don't loose sales

Sales: we can extend our offer basing on HA level

Accounts managers: we don't upset our customers (that often)

Developers: we can be proud – our services are working ;)

High Availability is in the eye of the beholder

CEO: we don't loose sales

Sales: we can extend our offer basing on HA level

Accounts managers: we don't upset our customers (that often)

Developers: we can be proud – our services are working ;)

System engineers: we can sleep well (and fsck, we love to!)

High Availability is in the eye of the beholder

CEO: we don't loose sales

Sales: we can extend our offer basing on HA level

Accounts managers: we don't upset our customers (that often)

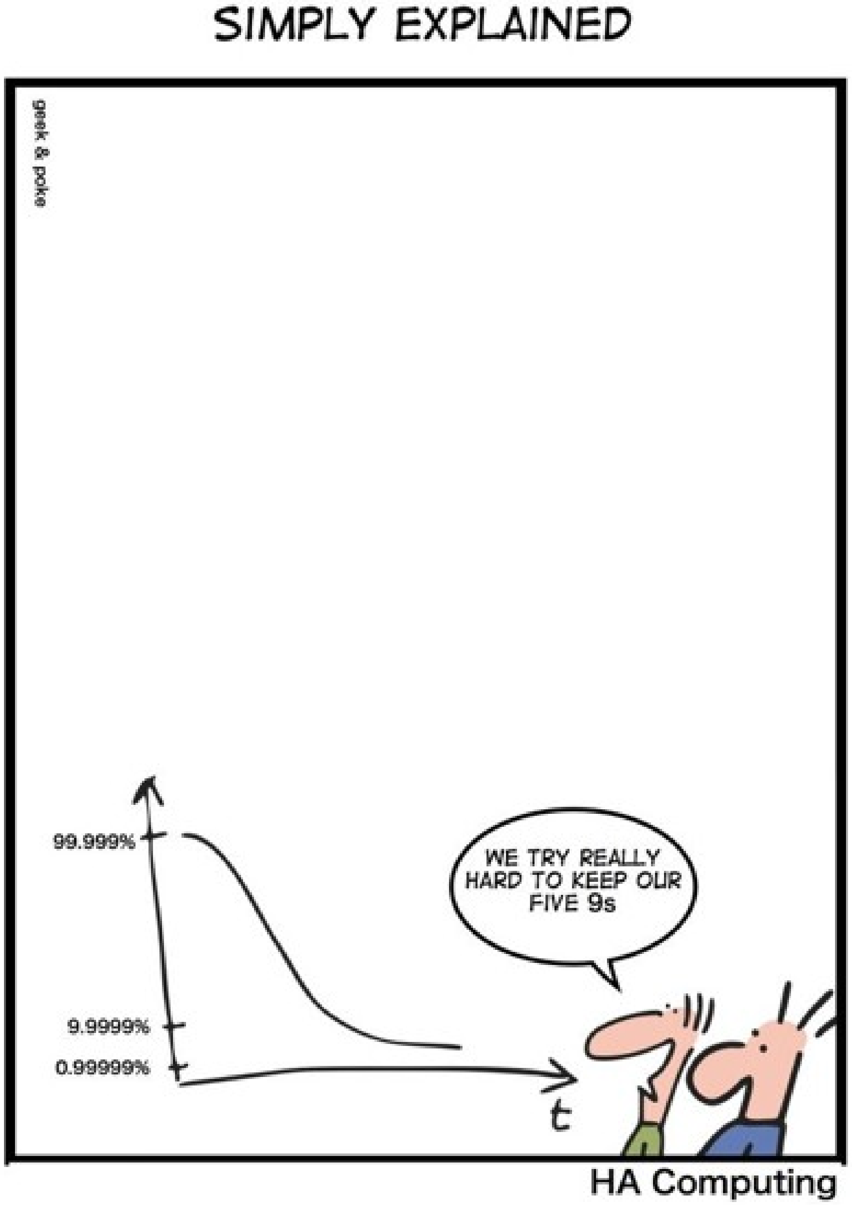
Developers: we can be proud – our services are working ;)

System engineers: we can sleep well (and fsck, we love to!)

Technical support: no calls? Back to WoW then.. ;)

So how many 9's?

So how many 9's?



So how many 9's?

Monthly: 1 hour of outage means 100% - 0.13888 ~= **99.86112** of availability

So how many 9's?

Monthly: 1 hour of outage means 100% - 0.13888 ~= **99.86112** of availability

Yearly: 1 hour of outage means 100% - 0.01142 ~= **99.98858** of availability

So how many 9's?

Monthly: 1 hour of outage means 100% - 0.13888 ~= **99.86112** of availability

Yearly: 1 hour of outage means 100% - 0.01142 ~= **99.98858** of availability

Availability	**Downtime (year)**	**Downtime (month)**
90% (“one nine”)	36.5 days	72 hours
95%	18.25 days	36 hours
97%	10.96 days	21.6 hours
98%	7.30 days	14.4 hours
99% (“two nines”)	3.65 days	7.2 hours
99.5%	1.83 days	3.6 hours
99.8%	17.52 hours	86.23 minutes
99.9% (“three nines”)	4.38 hours	21.56 minutes
99.99 (“four nines”)	52.56 minutes	4.32 minutes
99.999 (“five nines”)	**5.26 minutes**	**25.9 seconds**

So how many 9's?

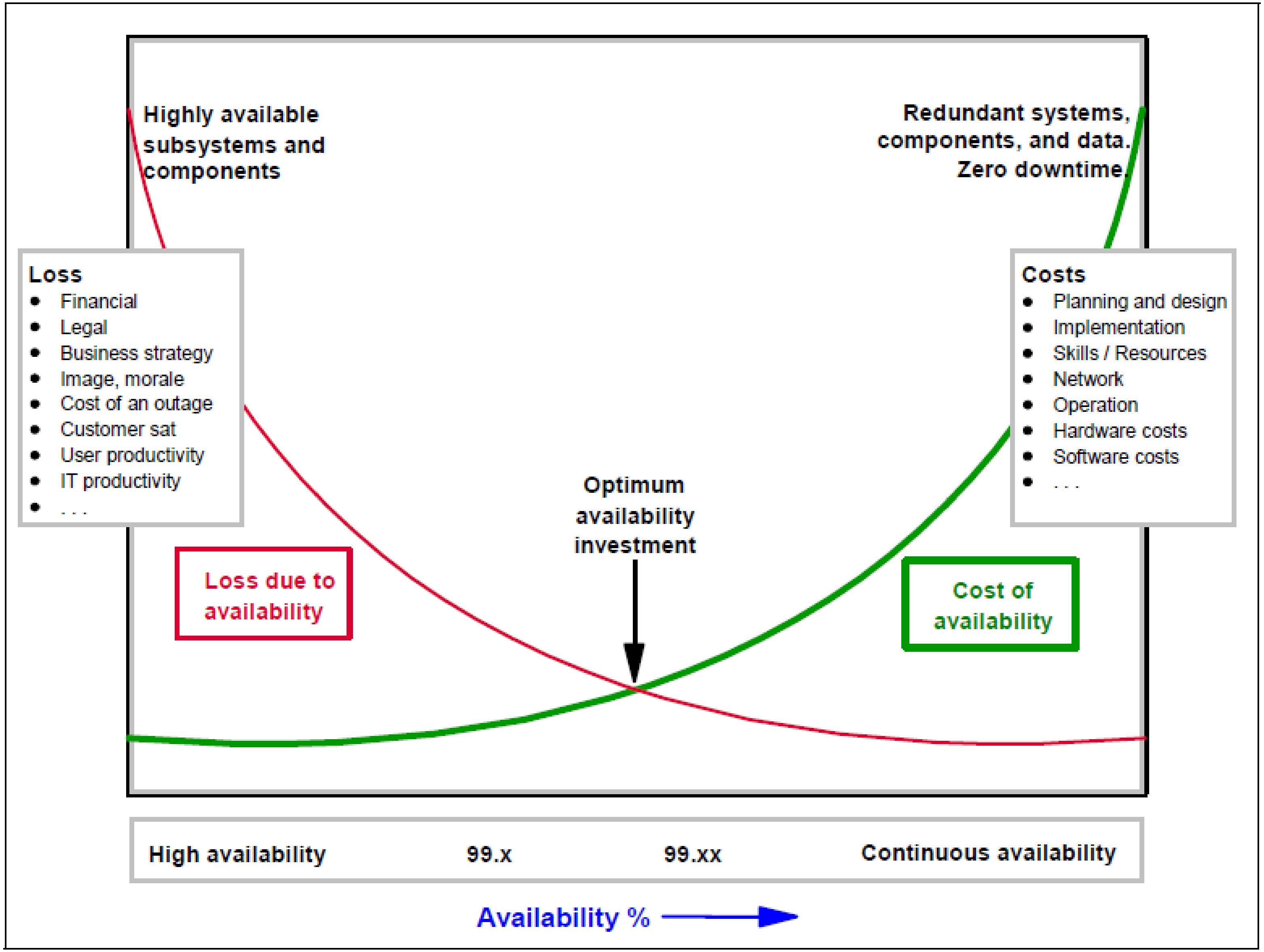


Figure 1-3   Cost of availability as opposed to loss due to availability

https://jazz.net/wiki/bin/view/Deployment/HighAvailability

HA terminology

RPO: Recovery Point Objective; how much data can we loose?

HA terminology

RPO: Recovery Point Objective; how much data can we loose?

RTO: Recovery Time Objective; how long does it take to recover?

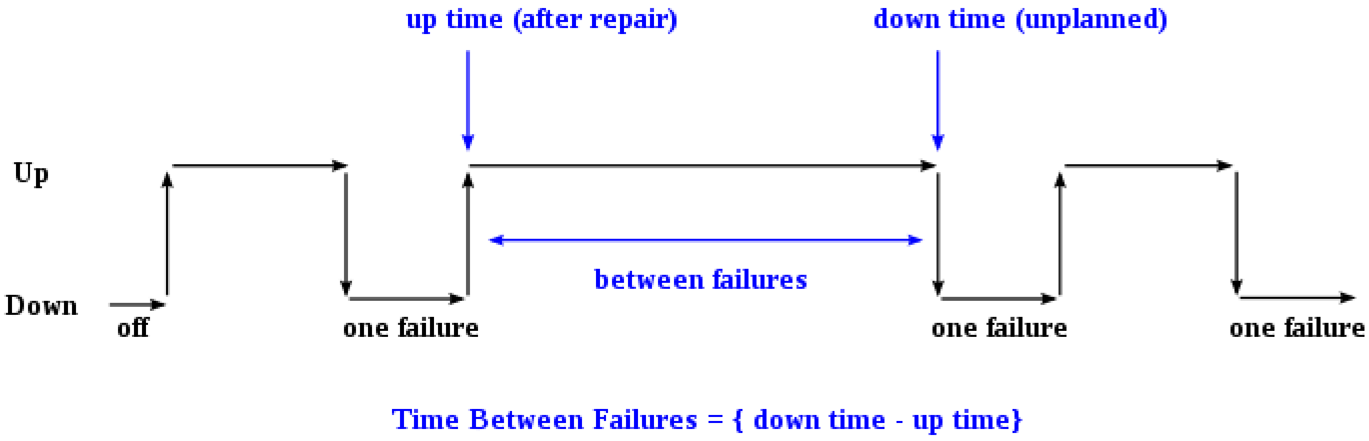
HA terminology

RPO: Recovery Point Objective; how much data can we loose?

RTO: Recovery Time Objective; how long does it take to recover?

MTBF: Mean-Times-Between-Failures; time between failures

(density fnc -> reliability fnc) $\int_0^\infty f(t)\,dt = 1.$



https://en.wikipedia.org/wiki/Mean_time_between_failures

HA terminology

SLA: Service Level Agreement;
formal definitions (customer <-> provider)

HA terminology

SLA: Service Level Agreement;
formal definitions (customer <-> provider)

OLA: Operational Level Agreement; definitions within organization;
help us keeping provided SLAs

SLAs..

So what is written in **SLAs**?

Availability	**Downtime (year)**	**Downtime (month)**
90%	36.5 days	72 hours
95%	18.25 days	36 hours
97%	10.96 days	21.6 hours
98%	7.30 days	14.4 hours
99%	3.65 days	7.2 hours
99.5%	1.83 days	3.6 hours
99.8%	17.52 hours	86.23 minutes
99.9%	4.38 hours	21.56 minutes
99.99	52.56 minutes	4.32 minutes
99.999	**5.26 minutes**	**25.9 seconds**

SLAs..

So what is written in **SLAs**?

Availability	**Downtime (year)**	**Downtime (month)**
90%	36.5 days	72 hours
95%	18.25 days	36 hours
97%	10.96 days	21.6 hours
98%	7.30 days	14.4 hours
99%	3.65 days	7.2 hours
99.5% **(EC2, EBS)**	1.83 days	3.6 hours
99.8%	17.52 hours	86.23 minutes
99.9% **(SoftLayer, IBM)**	4.38 hours	21.56 minutes
99.99 **(OVH ded. cloud)**	52.56 minutes	4.32 minutes
99.999	**5.26 minutes**	**25.9 seconds**

http://aws.amazon.com/ec2/sla/
http://www.softlayer.com/about/service-level-agreement
https://www.ovh.com/us/dedicated-cloud/security-and-sla.xml

SLAs..

Availability mentioned in **SLAs** are only **goals** of service provider

Usually when it's **not met** than company **pays off** the fees

SLAs..

Hetzner?

“We guarantee an annual average of 99% network availability”

“For indirect damages and loss of profits, we are liable only in cases of intentional or gross negligence. In this case we are liable only for the contract-typical predictable damage, a maximum of 100% of the annually fee.”

http://www.hetzner.de/en/hosting/legal/agb

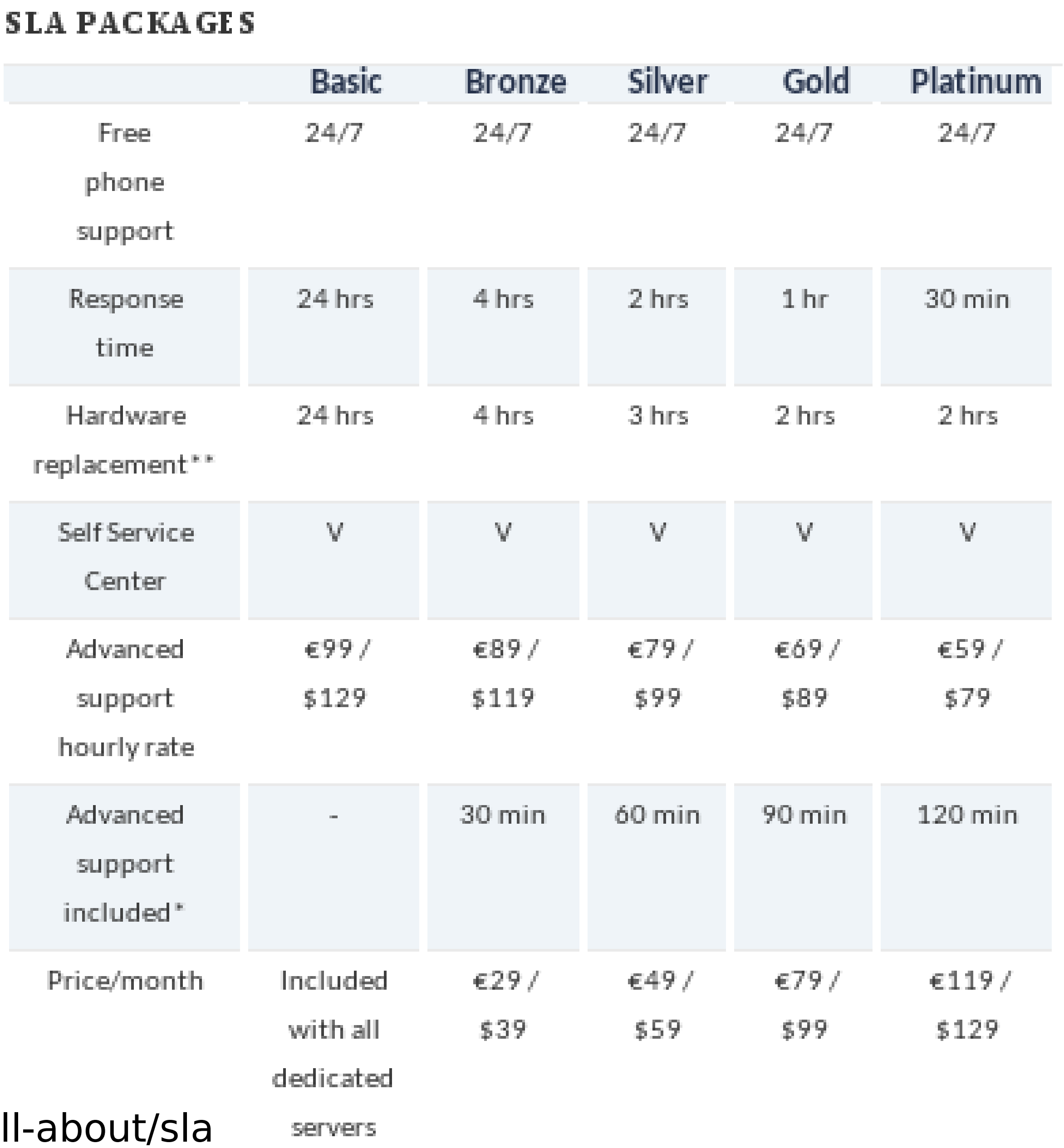
SLAs..

Leaseweb?
(yup - megaupload)

- no %s

- best effort

- $$$ for faster response times



SLA PACKAGES

	Basic	Bronze	Silver	Gold	Platinum
Free phone support	24/7	24/7	24/7	24/7	24/7
Response time	24 hrs	4 hrs	2 hrs	1 hr	30 min
Hardware replacement**	24 hrs	4 hrs	3 hrs	2 hrs	2 hrs
Self Service Center	V	V	V	V	V
Advanced support hourly rate	€99 / $129	€89 / $119	€79 / $99	€69 / $89	€59 / $79
Advanced support included*	-	30 min	60 min	90 min	120 min
Price/month	Included with all dedicated servers	€29 / $39	€49 / $59	€79 / $99	€119 / $129

http://www.leaseweb.com/en/support/all-about/sla

How deep is this hole?

app layer (core, db, cache)

data storage

operating system

hardware

networking

location

So we would like to achieve 99,9999% which is about 30s of downtime per year

How deep is this hole?

app layer (core, db, cache)

data storage

operating system

hardware

networking

location

Even Proof of Concept is very hard to provide: 2.5s of downtime per layer monthly!

How deep is this hole?

app layer (core, db, cache)

data storage

operating system

hardware

networking

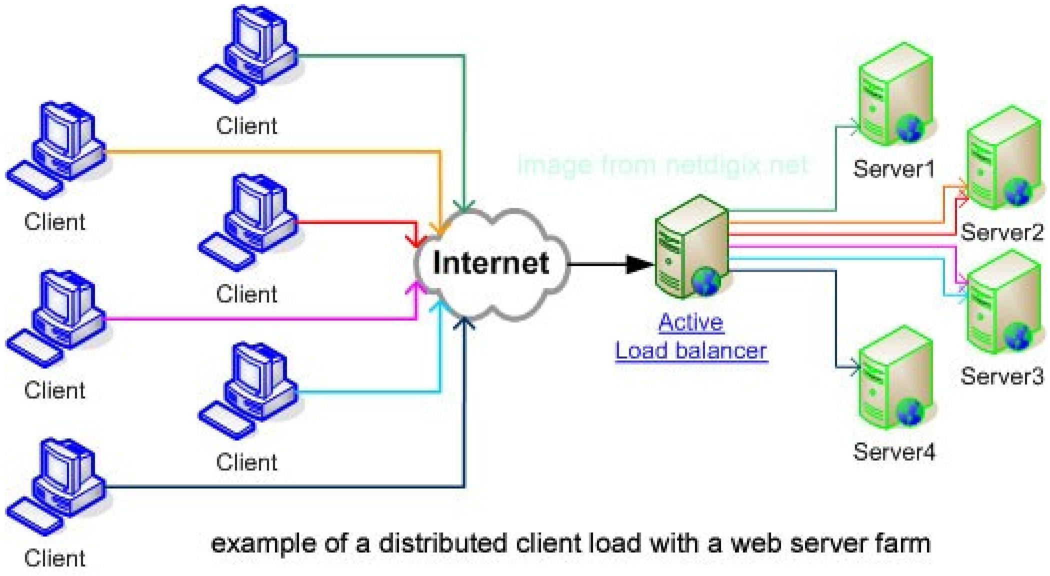
location

AOL has 99,999%!

http://highscalability.com/blog/2014/2/17/how-the-aolcom-architecture-evolved-to-99999-availability-8.html

Load-balancing and failover

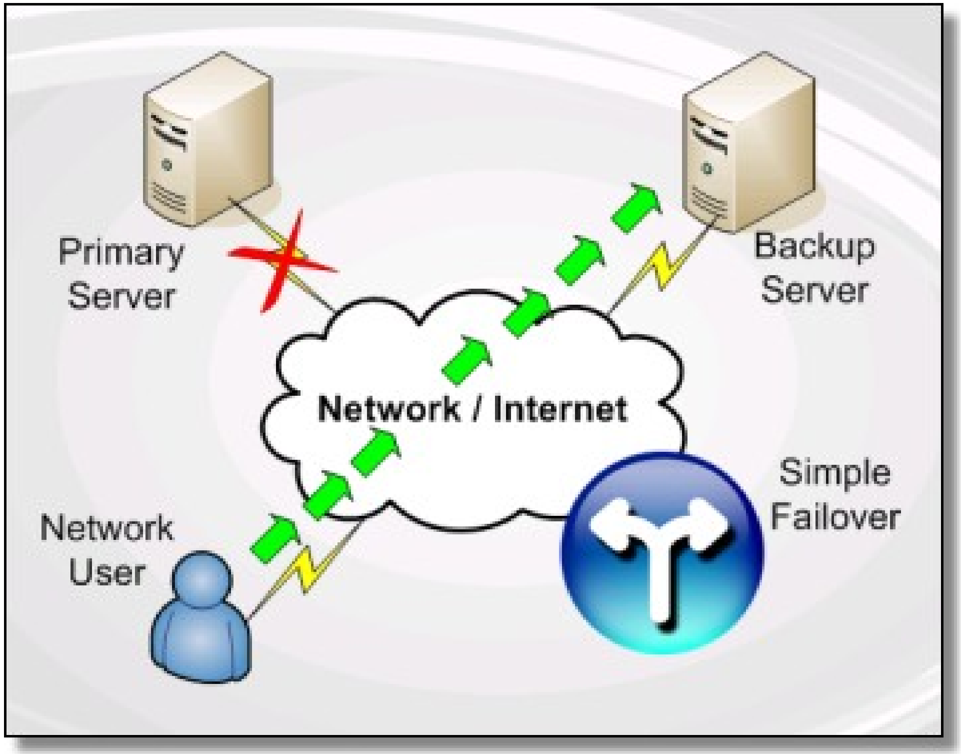
LB:



http://www.netdigix.com/linux-loadbalancing.php

Load-balancing and failover

Failover:



http://www.simplefailover.com/

LB – 4th layer or 7th?

4th layer:

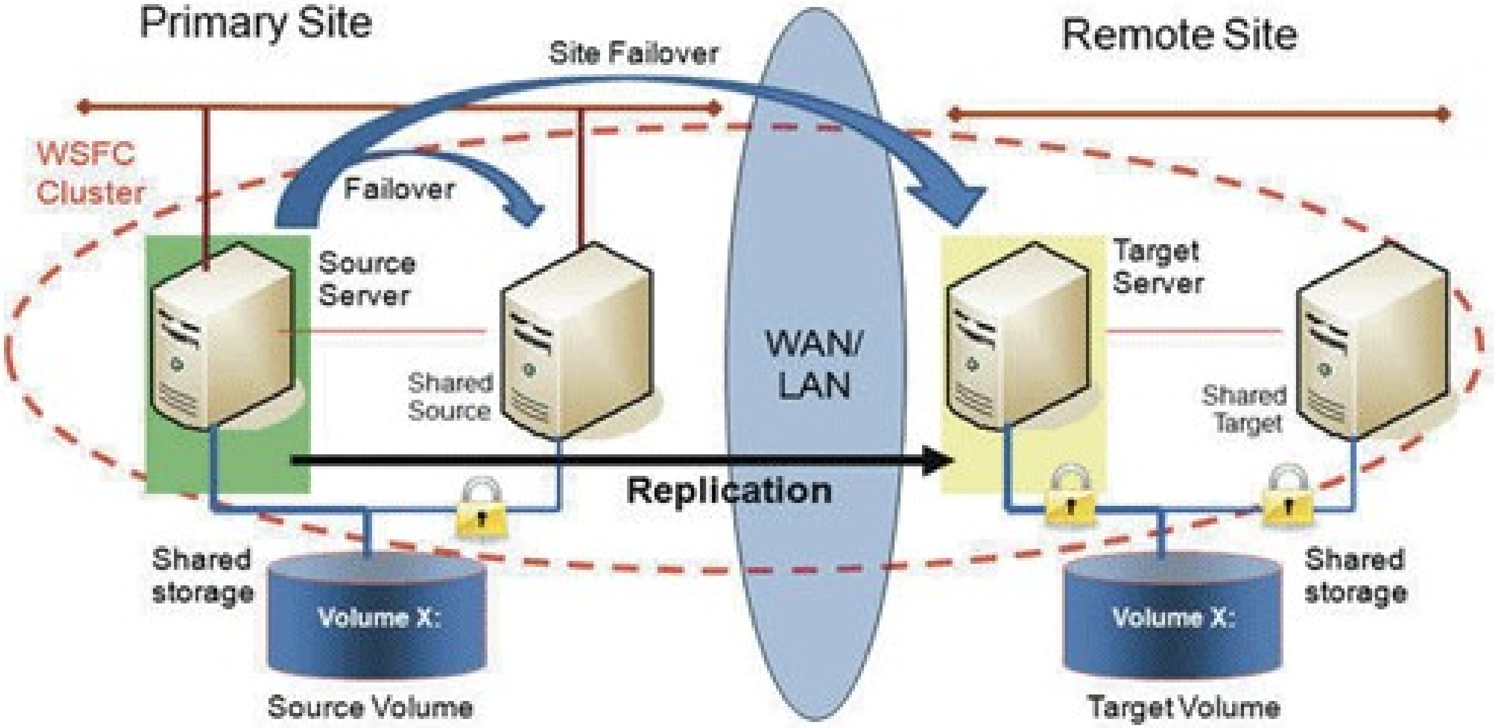
- high performance
- just do the LB work!
- reliable
- scalable

7th layer:

- low cost
- good for quickfixes / patches
- not that scalable
- low performance
- complex codebase
- custom code for protocols
- cookies? what about memcache..

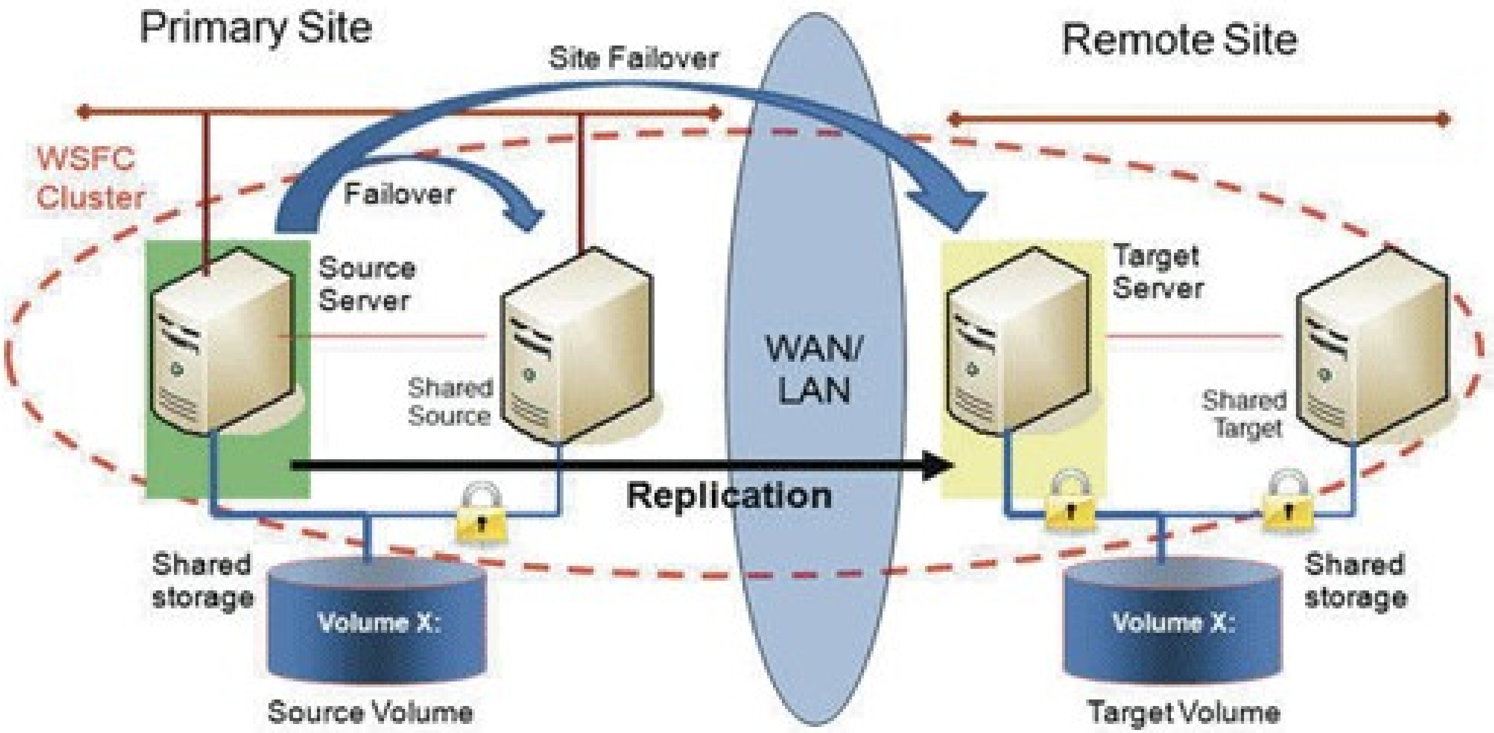
Disaster Recovery

Disaster Recovery



http://disasterrecovery.starwindsoftware.com/planning-disaster-recovery-for-virtualized-environments

Disaster Recovery



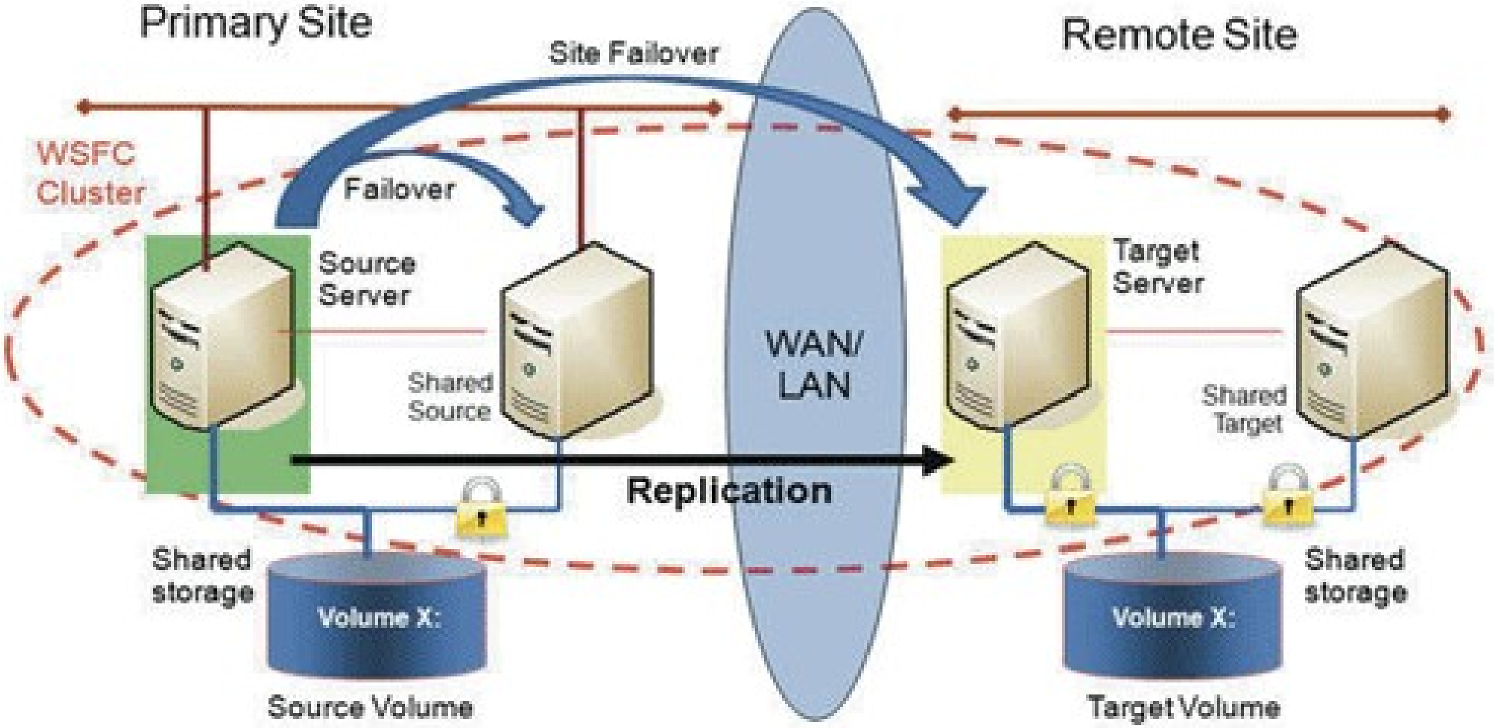
http://disasterrecovery.starwindsoftware.com/planning-disaster-recovery-for-virtualized-environments

Hot site: active synchronization, could be serving services. Cost can be high

Warm site: periodical synchronization, DR tests needed. Low costs

Cold site: Nothing here – just echo and some place to spin services; nightmare

Disaster Recovery

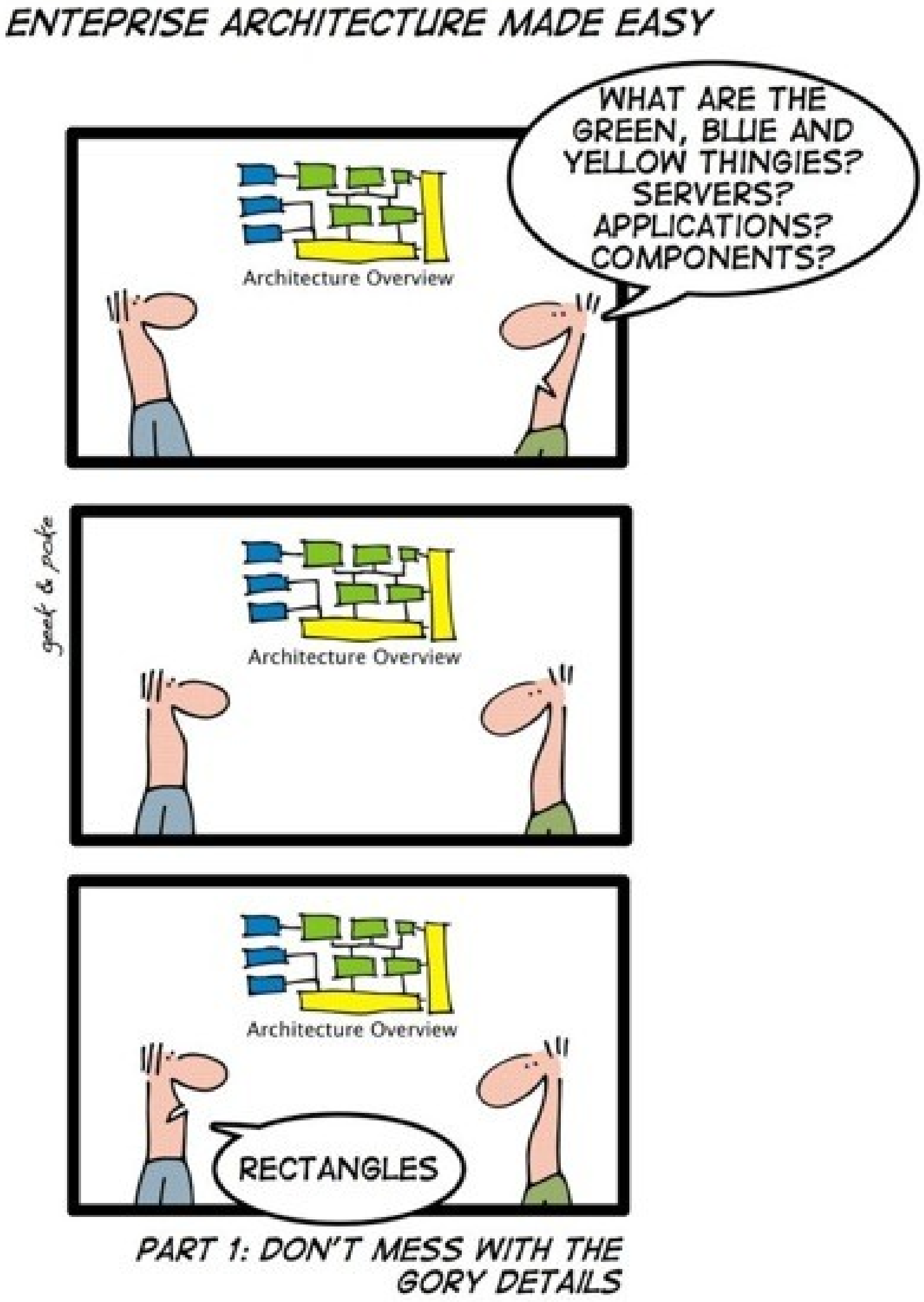


http://disasterrecovery.starwindsoftware.com/planning-disaster-recovery-for-virtualized-environments

Hot site: active synchronization, could be serving services. Cost can be high

So maybe use “hot” as production and global LB? win-win scenario

Planning for failure


ENTEPRISE ARCHITECTURE MADE EASY
Architecture Overview
WHAT ARE THE GREEN, BLUE AND YELLOW THINGIES? SERVERS? APPLICATIONS? COMPONENTS?
geek & poke
Architecture Overview
Architecture Overview
RECTANGLES
PART 1: DON'T MESS WITH THE GORY DETAILS

Planning for failure

Everything starts here - DNS:

- keep TTLs low (300s). Can't make under 60min? That's bad!
- check SLA of DNS servers (dnsmadeeasy.com history)
- what do you know about DNSes?
    - zero downtime here is a must!
    - this can be achieved with complicated network abracadabra
    - remember what 99.9999% means?
- round robin is a load – balancer but **without failover**!
- GSLB – killed by OS/browser/srvs cache'ing
  (GlobalServerLoadBalancing)
- GlobalIP (SoftLayer etc) – workaround for GSLB via routing

Planning for failure

E-mail servers:

- it's simple as MX records (**delivering**)

- it's almost simple as complicated system of SMTP servers (**sending**)

- it's not that simple when IMAP locking over DFS (**reading**)

```
5 gmail-smtp-in.l.google.com.
10 alt1.gmail-smtp-in.l.google.com.
20 alt2.gmail-smtp-in.l.google.com.
30 alt3.gmail-smtp-in.l.google.com.
40 alt4.gmail-smtp-in.l.google.com.
```

When MXing – **watch the spam**!

Planning for failure

WEB servers:

- it's simple as some frontend loadbalancer

- did you **really** stick user session to particular server? Memcache!

- LB balancing algorithm

- how many LBs?

- what if LB goes down?

Planning for failure

DB servers:

- it's.. not that simple
- replication (master – master? App should be aware..)
- replication ring? Complicated, works, but in case of failure...
- let's talk about MySQL:
  - NoSPOF solution: MySQL cluster
  - MySQL Galera cluster – synch, active-active multi-master
  - master – master – simply works
  - MySQL fabric – HA + sharding; use with large farms
  - Failover? Matsunobu Yoshinori mysql-master-ha
  - MySQL utilities (http://www.clusterdb.com/mysql/mysql-utilities-webinar-qa-replay-now-available/)

Planning for failure

DB servers:

Matsunobu Yoshinori mysql-master-ha

https://code.google.com/p/mysql-master-ha/

“I have heard a couple of cases where AWS users use MHA instead of RDS because RDS takes much longer downtime on slave promotion (more than 5 minutes usually, because standby database is not running).

I'm surprised AWS users care about a few minutes of downtime..”

Planning for failure

Caching servers:

- this is cache for God's sake – why would we use HA here?

- just use proper architecture like... redundancy.

Planning for failure

Caching servers:

- this is cache for God's sake – why would we use HA here?

- just use proper architecture like... redundancy.

Load – balancers:

- remember about failovering IP addresses!

Planning for failure

Caching servers:

- this is cache for God's sake – why would we use HA here?

- just use proper architecture like... redundancy.

Load – balancers:

- remember about failovering IP addresses!

Storage – DFSes:

- GlusterFS – we'll see it in action in a minute

- NFS? Could be – over some SAN / NAS (high cost solution)

- CephFS – just like GlusterFS – it's great and does the work

- DRBD – lower level, does the work on block – device layer – slow...

Planning for failure

GlusterFS:

- low cost (could be..)
- distributed volumes
- replicated volumes
- striped volumes
- and...
  - distributed – striped volumes
  - distributed – replicated volumes
  - distributed – striped – replicated volumes
- sound good? :)

Planning for failure

GlusterFS: replicated volumes vs Geo-replication

- replicated:
  - mirrors data
  - provides HA
  - synch – replication
- Geo-replication:
  - mirrors data across geo – distributed clusters
  - ensures backing up data for DR
  - asynch – replica (periodic checks)

Planning for failure

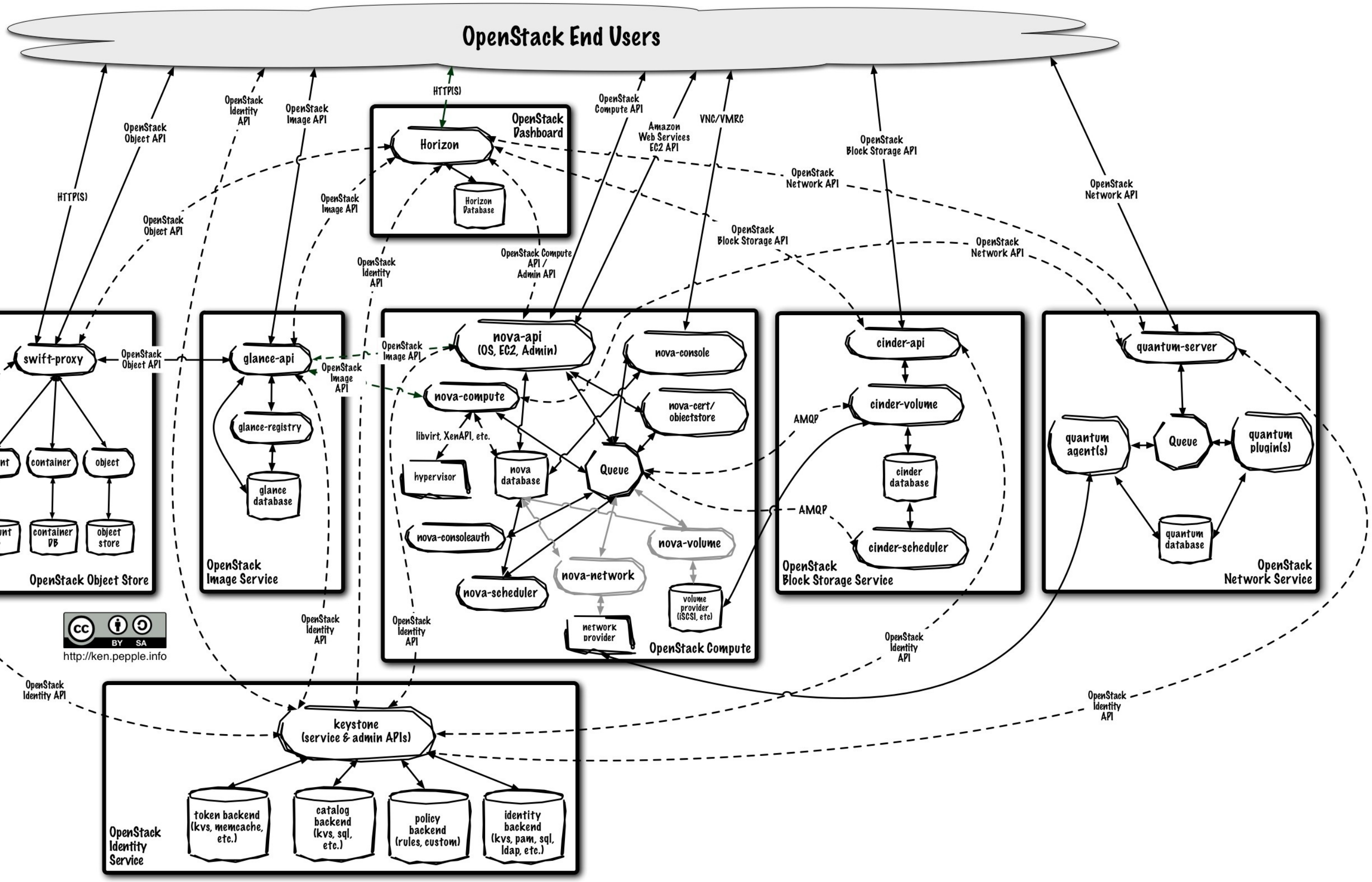
HA for virtualization solutions?

- it's really complicated, like...

Planning for failure

HA for virtualization solutions?

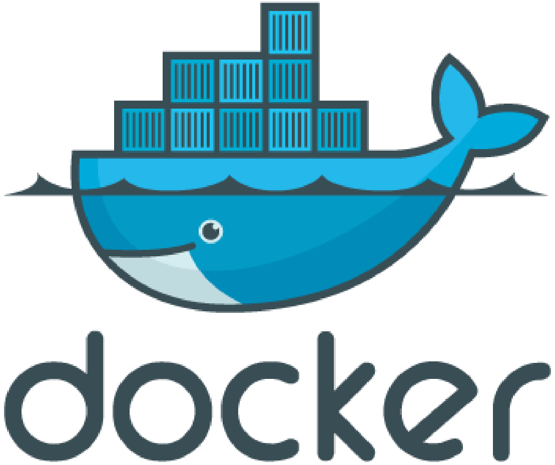
- it's really complicated, like...



Planning for failure

HA for virtualization solutions?

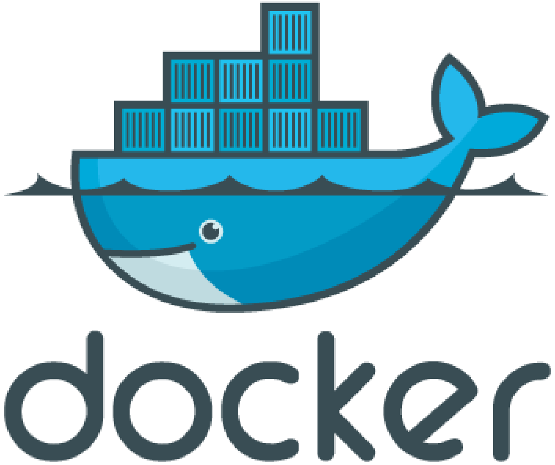
- but it could be done simpler...



Planning for failure

HA for virtualization solutions?

- it could be done simpler with containers!
- containers are very light
- deploy time from bare is tiny
- management is very easy
- resources throttling via cgroups

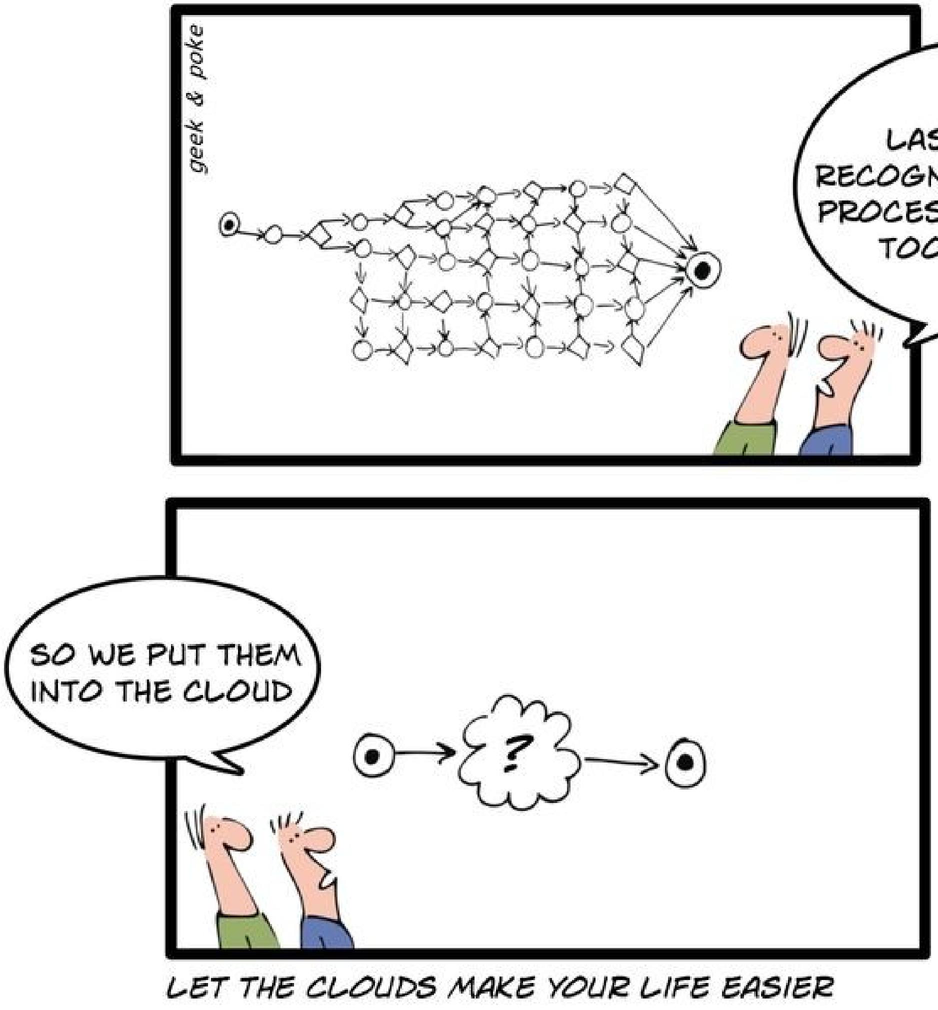


Tools

The most important tool would be the conclusion from the picture below:

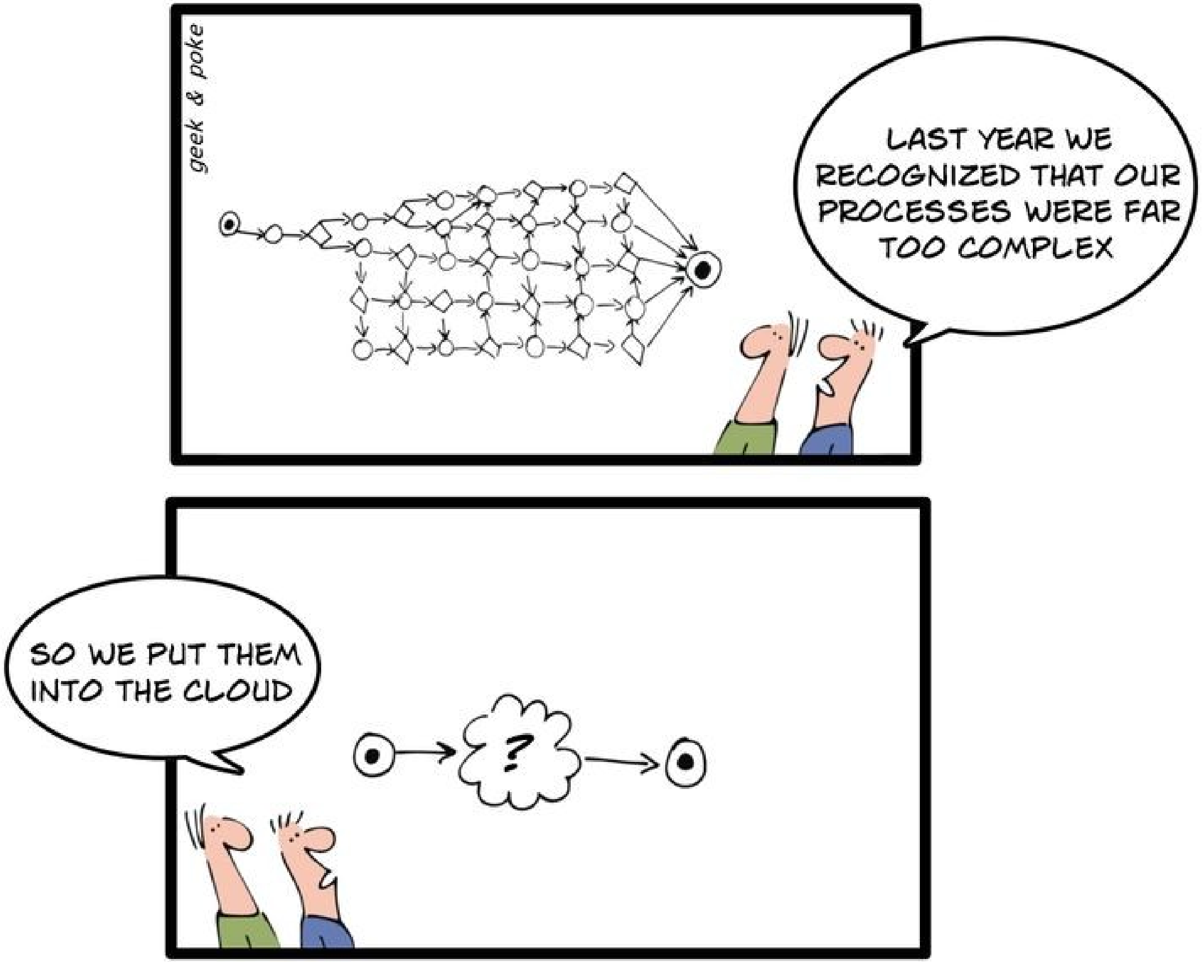
Tools

The most important tool would be the conclusion from the picture below:


geek & poke
LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX
SO WE PUT THEM INTO THE CLOUD
?
LET THE CLOUDS MAKE YOUR LIFE EASIER

Tools

The most important tool would be the conclusion from the picture below:


geek & poke
LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX
SO WE PUT THEM INTO THE CLOUD
?

LET THE CLOUDS MAKE YOUR LIFE EASIER

Tools

- DNS: roundrobin, GSLB, low ttls, globalIP

Tools

- DNS: roundrobin, GSLB, low ttls, globalIP

- Load-Balancers (l7, stateless services)): HaProxy, Pound, Nginx

Tools

- DNS: roundrobin, GSLB, low ttls, globalIP
- Load-Balancers (l7, stateless services)): HaProxy, Pound, Nginx
- Failover (statefull services):
    - IP: KeepAlived + sysctl

Tools

- DNS: roundrobin, GSLB, low ttls, globalIP
- Load-Balancers (l7, stateless services)): HaProxy, Pound, Nginx
- Failover (statefull services):
    - IP: KeepAlived + sysctl
    - Managing: pacemaker (manager) + corosync (message'ing)

Tools

- DNS: roundrobin, GSLB, low ttls, globalIP

- Load-Balancers (l7, stateless services)): HaProxy, Pound, Nginx

- Failover (statefull services):

    - IP: KeepAlived + sysctl

    - Managing: pacemaker (manager) + corosync (message'ing)

- (almost) All-In-One: Linux Virtual Server

Turn on HA thinking!

Main goal of HA? **Improve user experience**!

- keep the app **fully functional**

- keep the app **resistant and tolerant** to faults

- provide method for a **successful audit**

- **sleep well** (anyone awake?) ;)

Thank you :)

High Availability Explained

Maciej Lasyk

11. Sesja Linuksowa

2014-04-05, Wrocław

http://maciek.lasyk.info/sysop

maciek@lasyk.info

@docent-net