HowTo DR

Help!

Josh Berkus
PostgreSQL Experts
SCALE 2014
Disaster Recovery

“The process, policies and procedures that are related to preparing for recovery or continuation of technology infrastructure which are vital to an organization after a natural or human-induced disaster.”

Wikipedia, February 2014
Disaster Recovery

Restoring services after the unexpected.
Disaster Recovery

Limiting:
1. Downtime
2. Data Loss
Do you have a DR Plan?
Is it fairly complete?
Have you **tested** it?
Our Disaster Recovery Plan
Goes Something Like This...

HELP!
HELP!

Dilbert is a copyright of Scott Adams. Used here as parody. All rights reserved to Scott Adams.
Threat Model
server
failure

going
hacked

natural
disaster
server failure
storage failure
getting hacked
natural disaster
network failure
traffic spike
admin error
bad update
OS / VM problem
software bugs
Accepting Loss
<table>
<thead>
<tr>
<th>Nines</th>
<th>Down/Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.9%</td>
<td>9 hours</td>
</tr>
<tr>
<td>99.99%</td>
<td>1 hour</td>
</tr>
<tr>
<td>99.999%</td>
<td>5 minutes</td>
</tr>
</tbody>
</table>
The Nines

- Treats all downtime causes as identical
  - except the ones it ignores
- Doesn't address data loss
- Really “Business Continuity”
- also unrealistic
<table>
<thead>
<tr>
<th>Disaster</th>
<th>Downtime</th>
<th>Data Loss</th>
<th>Detect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admin Error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bad Update</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Getting Hacked</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Natural Disaster</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disaster</td>
<td>Downtime</td>
<td>Data Loss</td>
<td>Detect</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>Server Failure</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network Failure</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Admin Error</td>
<td>0</td>
<td>0</td>
<td>10 yrs</td>
</tr>
<tr>
<td>Bad Update</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Storage Failure</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Getting Hacked</td>
<td>0</td>
<td>0</td>
<td>10 yrs</td>
</tr>
<tr>
<td>Natural Disaster</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Disaster</td>
<td>Downtime</td>
<td>Data Loss</td>
<td>Detect</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>Server Failure</td>
<td>5min</td>
<td>1min</td>
<td></td>
</tr>
<tr>
<td>Network Failure</td>
<td>3hrs</td>
<td>10min</td>
<td></td>
</tr>
<tr>
<td>Admin Error</td>
<td>1hr</td>
<td>1hr</td>
<td>3 mo</td>
</tr>
<tr>
<td>Bad Update</td>
<td>1hr</td>
<td>1hr</td>
<td></td>
</tr>
<tr>
<td>Storage Failure</td>
<td>5min</td>
<td>30min</td>
<td></td>
</tr>
<tr>
<td>Getting Hacked</td>
<td>1hr</td>
<td>1hr</td>
<td>3 mo</td>
</tr>
<tr>
<td>Natural Disaster</td>
<td>6hrs</td>
<td>1hr</td>
<td></td>
</tr>
</tbody>
</table>
Your DR Plan
Elements of a Plan

1. Backups/Replicas
2. Replacements
3. Procedures
4. People
Backups

- pg_dump
- mysqldump
- rsync
- zfs snapshot
- SAN exportable snapshot
Backups++

- Periodic
- Portable
- Simple
- Recover point-in-time
Backups--

- Slow to restore
- Data loss interval
Backups

- Good for:
  - natural disaster
  - admin error, bad update
  - software bugs
  - getting hacked

- Bad for everything else
Replication

- Postgres binary replication
- MySQL replication
- Redundant HBase nodes
- Redis clustering
- DRBD
- GlusterFS etc.
Replication++

- Continuous
- Fast failover
- Low data loss
Replication--

- Extra hardware
- Complex
- High-maintenance
- Can hurt performance
- Can replicate failures
Replication

- **Good For:**
  - server, storage, network failure
- **Bad For:**
  - admin error, getting hacked
  - software bugs
Continuous Backup

- Also “PITR”
- Continuous like replication
- Partial recovery like backups
... where you gonna restore those backups to?
Replacing Services

- servers
- network
- storage
- OS image
- software reversion
Written Procedures
3AM is not the time to improvise
Procedures

... for each recovery step
... for deciding what steps
Database Server
Does Not Respond

1. Determine if physical server is down
   a. If network is down, use plan N1.
2. If not, try to restart database using command ...
3. Still down? Fail over to replica using command ...
4. Check replica.
5. Not working? Restore backup to test server 1 using command ...
Good: detailed written procedures

Better: written procedures with pastable commands

Best: tested single-command scripts
Fallback Procedures

- Sometimes recovery fails
- Have fallback procedures
- If the fallback fails
  ... time for a meeting!
Never ever ever ever ever improvise.
People

Who You Gonna Call?
Know who to call

- on call staff
- experts in each service
- consultants/contractors
- vendors
- required authorizations
Contact Book

- Include as much contact information as possible
- Put copies in more than one place
  - including paper!
- Keep it up to date
Test Your DR

**Good**: when you create the procedure

**Better**: quarterly

**Best**: as part of daily/weekly provisioning
An untested backup is one which doesn't work.
DR in the Cloud
“It's a cloud, right? That means it's redundant, right?”
... not necessarily for your servers

<table>
<thead>
<tr>
<th>Type</th>
<th>State</th>
<th>Status Checks</th>
<th>Monitoring</th>
<th>Security Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.large</td>
<td>running</td>
<td>✋ 0/2 checks passed</td>
<td>basic</td>
<td>default</td>
</tr>
</tbody>
</table>

Instance reachability check failed. System reachability check failed.

unless you pay for it!
Some new problems

- Instance failure
- Resource overcommit
- Zone failures
- Admin error at scale
Some new solutions

- Redundant services
  - RDS, VIP, S3
- Rapid server deployment
- Cheap replicas
... otherwise pretty much the same.
backup locations

- shared instance storage (EBS)
  - fast failover for instance fail
- long-term storage API (S3)
  - redundant
  - large
Use your rapid deploy!

- Continuous backup to S3
- Deploy scripts + server images
  - Chef/Salt/Puppet/etc. helps here
- = fast recovery
  - with low running costs
DR Tips

- Have multiple copies of your plan
  - in multiple locations
- A SAN is not a DR solution
- One form of backup is seldom enough
Questions?

- Josh Berkus
  - www.databasesoup.com
  - www.pgexperts.com

- Coming up:
  - NYC pgDay April 3-4
  - pgCon May 21-24