# PostgreSQL on ZFS

Replication, Backup, Human-disaster Recovery, and More.

Keith Paskett

# Agenda

1. Introductions
2. Database challenges that ZFS alleviates
3. ZFS/OpenIndiana overview and practice
4. Database replication & backup via ZFS
5. Cloning for validation, testing and recovery
6. ZFS and point-in-time recovery
7. ZFS with file/streaming replication
8. Conclusion
9. Q&A

# Introductions

1. Name
2. A little about yourself
3. Postgres experience level
4. Unix/Linux command line comfort level
5. What you expect to get out of this tutorial

# Backup & Recovery Disconnect

- Less likely disaster scenarios
  - Server failure
  - Multiple simultaneous drive failures
  - Data center flattened by (choose your disaster)
- Common 'human' disaster scenarios
  - Dropped table
  - Deleted data
  - Altered data
- Many backup solutions focus on the less likely

# I Wish I Could...

- Test an upgrade script on 2TB database without having to set up a server with 2TB
- Quickly roll back from an upgrade if things go badly
- Have point-in-time access to a large database without having to do a restore

# ZFS Advantages for Databases

- Fast efficient replication
- Low/No-impact snapshots
- Read/write access to snapshots via clones
- Pool physical devices
- Bidirectional incremental send/receive
- Solid State cache drives.
- Upgrade to larger physical drives with 0 downtime
- Continuous integrity checking and automatic repair

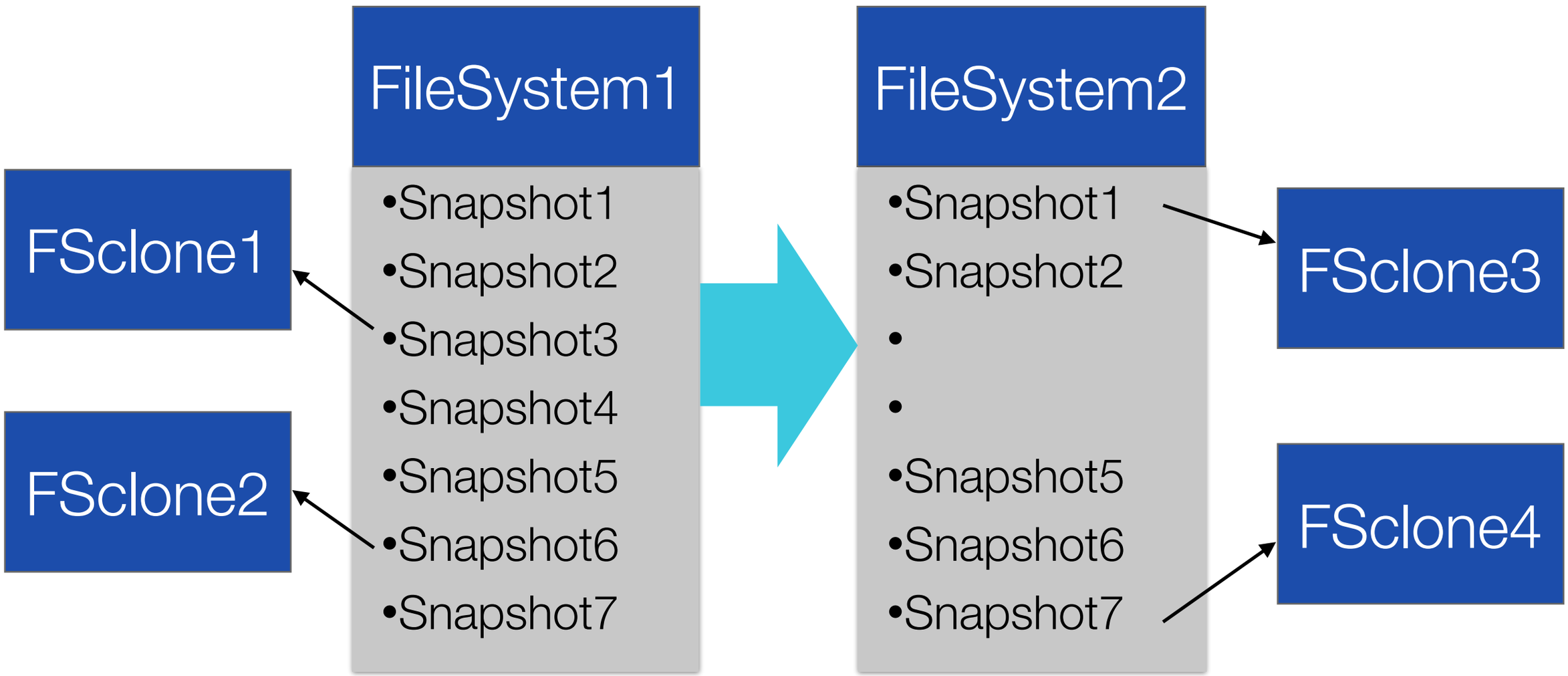
# VM setup and practice

VM available at http://static3.usurf.usu.edu/pgopen-zfs.ova

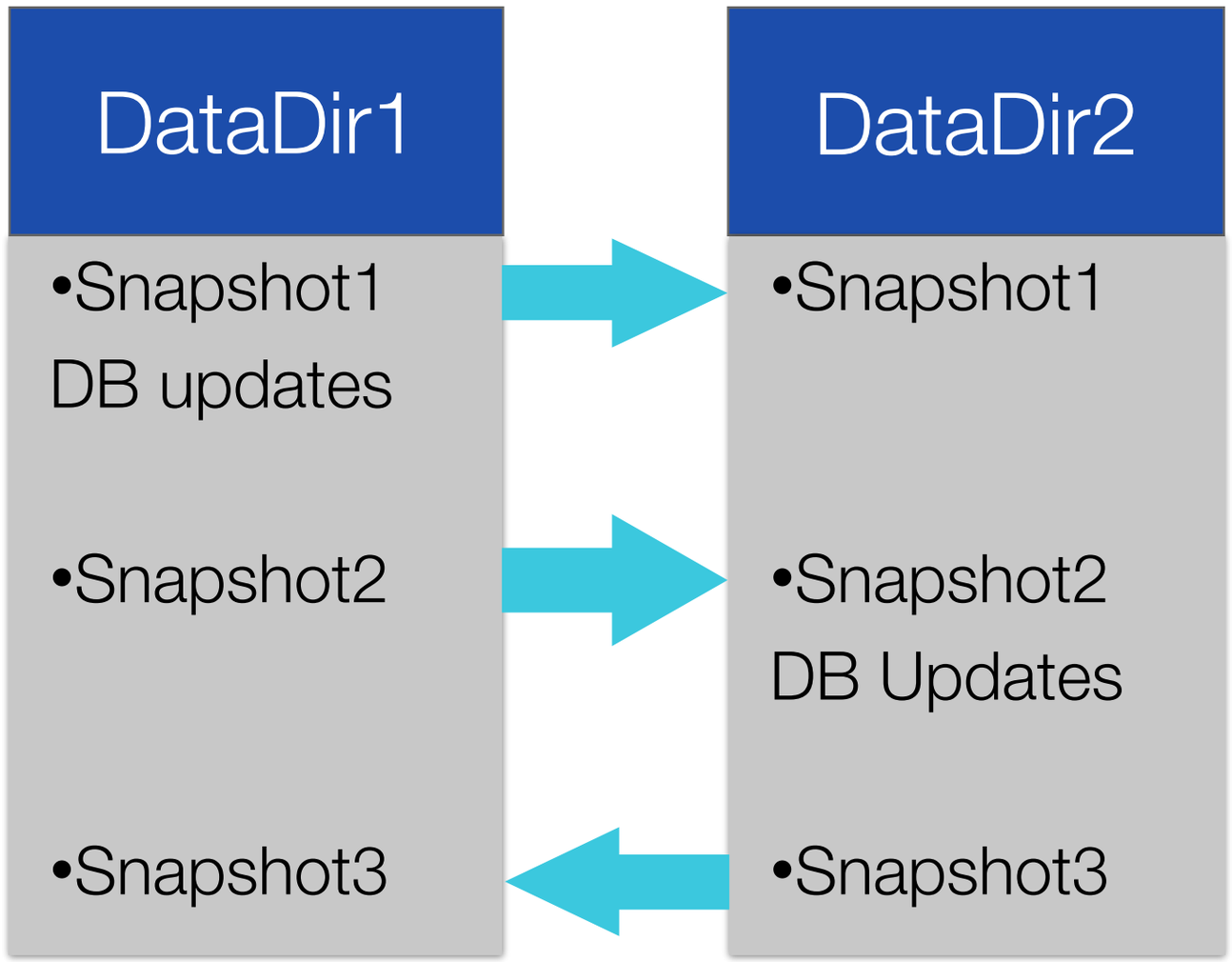
# VM Setup and Practice

- Import the OpenIndiana VM start it up and log in:
  username/password: admin/nimda
- mv .bin bin then restart your terminal
- Start up all zones: pfexec bin/start-zones.sh
- Create the filesystem: rpool/myfs
- Add some content: cp -r Downloads /rpool/myfs/
- Create a snapshot: zfs snapshot rpool/myfs@rep1
- Add more content:  cp -r Documents /rpool/myfs/
- Replicate from the first snapshot:
  zfs send rpool/myfs@rep1 | zfs recv rpool/myfs-copy@rep1
- Destroy both new filesystems:
  zfs destroy -r rpool/myfs;  zfs destroy -r rpool/myfs-copy

# ZFS - It's all about the snapshots



# Exercise 1 - Send & Receive

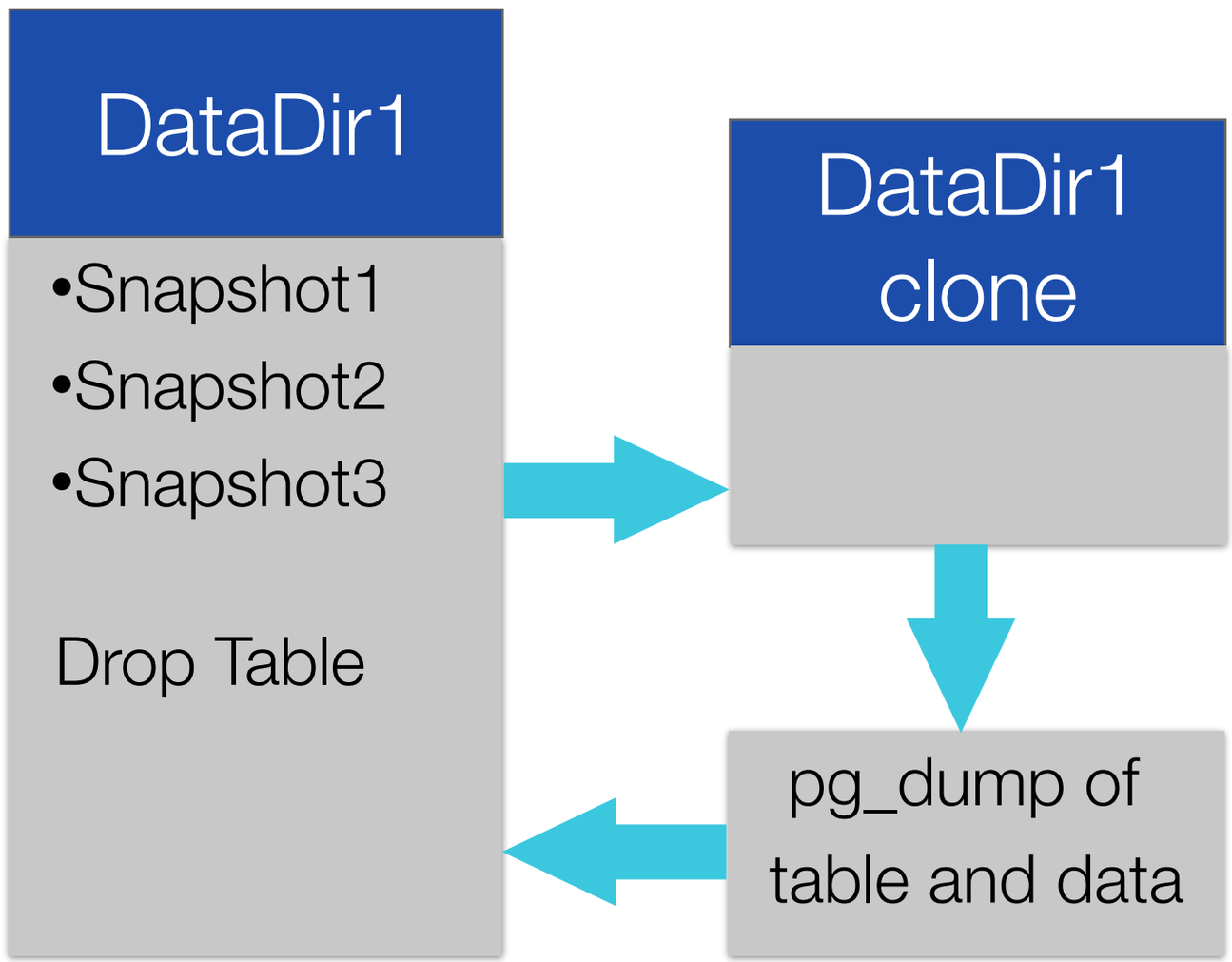



"ls Documents" to see files with exercize steps

# Accessing Snapshot Contents

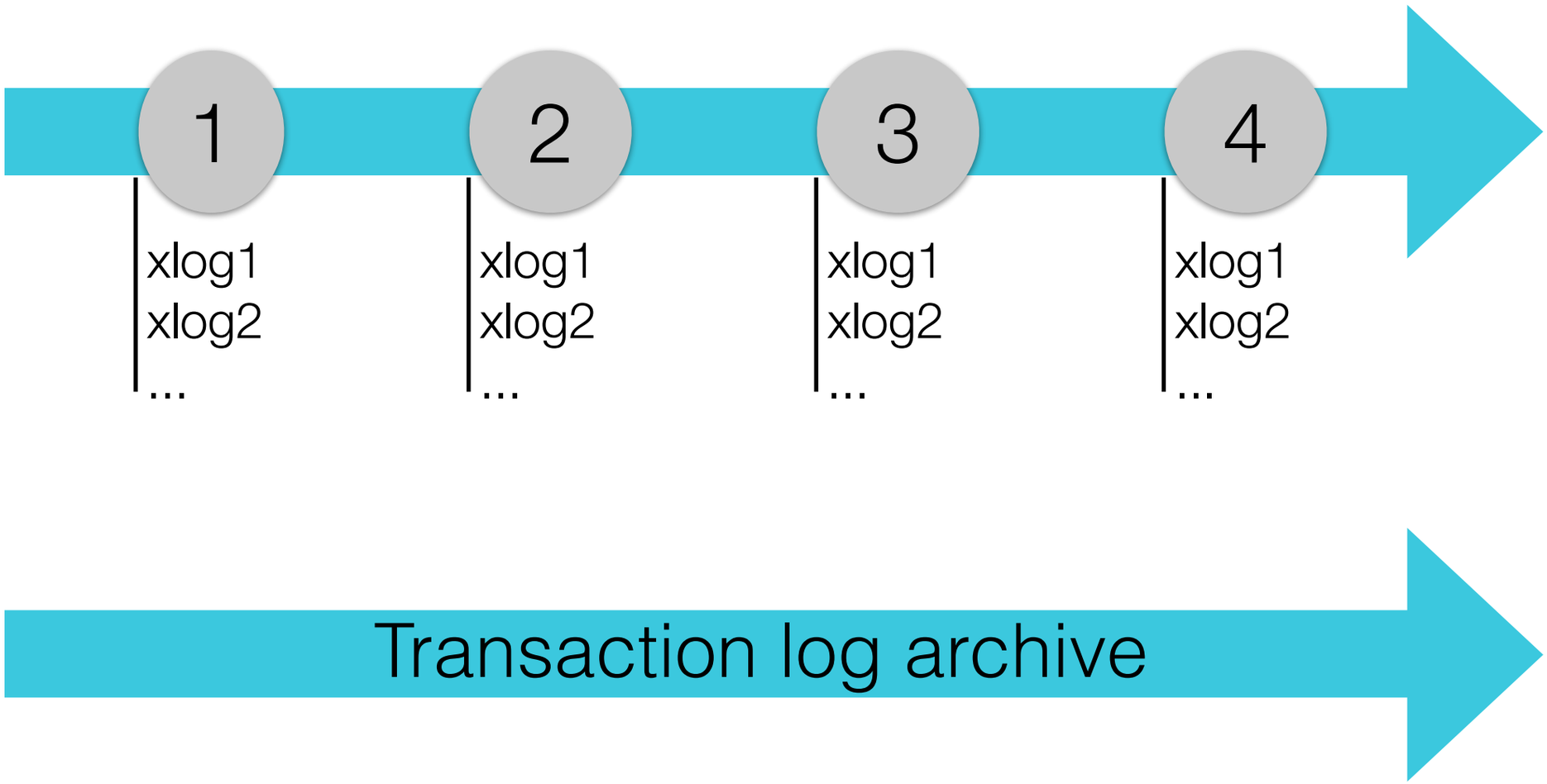
- .zfs directory
  - 'cd .zfs/snapshot/snapname' from base ZFS directory
  - 'ls -a' won't show the .zfs directory
  - Snapshot directories are read only
  - Tape backup of snapshot directory will be consistent
- Create a clone from a snapshot
  - zfs clone fsname@snapshotname clonename
  - Clones are writable
  - Only take additional disk space as data is changed
  - Simply awesome for replicating a TB database in seconds

# Exercise 2 - Clone & Recover




"ls -a Documents" to see files with exercize steps

# Snapshots + Point-in-time Recovery.



# Snapshot Frequency

- Tune to your needs
- My preference for transactional databases
  - Every 10-20 minutes, keep for 9 hours
  - Daily, keep for 10 days
  - Weekly, keep for 8 weeks
- Cronable script in admin's bin directory

# The OpenIndiana Virtual Machine

## Global zone

- The /shared1 and /shared2 directories in the global zone are the same as /shared1 and /shared2 in zones 1-4
- Each zone has its own isolated postgres instance
- Zones are connected on a local network using the 192.168.127.x subnet

**Zone1**

**File-based Master**

**Zone2**

**File-based Slave**

**Zone3**

**Streaming Master**

**Zone4**

**Streaming Slave**

# The OpenIndiana VM Zones

- Become all-powerful: pfbash
- Start host only network: bin/start-network.sh
- List zones: zoneadm list -cv
- Boot a zone: zoneadm -z zonename boot
- Boot all zones: pfexec bin/start-zones.sh
- Log in to a zone: pfexec zlogin -l postgres zonename
- Start/Stop postgres in a zone (as user postgres):

  bin/start-pg.sh and bin/stop-pg.sh
- View/edit zone configuration: zonecfg -z zonename
- Scripts to launch zone-to-zone Postgres replication are in admin's bin directory

# Summary & Caveats

- Great for quick duplication of very large databases
- Very efficient for periodic updates of replicas
- Combine snapshots with transaction log archives for faster access to point-in-time.
- Disk space only freed when files and snapshots deleted
- Have to shut down the 'receiving' DB during the update
- Linux port in progress (not production ready)

# Thank you.

Keith Paskett