

# Efficiently Backing up Terabytes of Data with pgBackRest

David Steele  
Crunchy Data

PostgresOpen 2016  
September 14, 2016



# Agenda

- 1 Why Backup?
- 2 Living Backups
- 3 Design
- 4 Features
- 5 Performance
- 6 Demonstration
- 7 Questions?

# Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is of course a challenge.

# Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is of course a challenge.

# Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is of course a challenge.

# Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is of course a challenge.

# Why Backup?

- Hardware Failure:
  - No amount of redundancy can prevent it.
- Replication:
  - WAL archive for when async streaming gets behind.
  - Sync replica from backup instead of master.
- Corruption:
  - Can be caused by hardware or software.
  - Detection is of course a challenge.

# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.



# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

# Why Backup?

- Accidents:
  - So you dropped a table?
  - Deleted your most important account?
- Development:
  - No more realistic data than production!
  - May not be practical due to size / privacy issues.
- Reporting:
  - Use backups to standup an independent reporting server.
  - Recover important data that was removed on purpose.

# Schrödingers Backup

The state of any backup is unknown until a restore is attempted.

# Making Backups Useful

- Find a way to use your backups
  - Syncing / New Replicas
  - Offline reporting
  - Offline data archiving
  - Development
- Unused code paths will not work when you need them unless they are tested
  - Regularly scheduled automated failover using backups to restore the old primary
  - Regularly scheduled disaster recovery (during a maintenance window if possible) to test restore techniques

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.



# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
    - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Design

- Rsync powers many database backup solutions but it has some serious limitations:
  - Single-process.
  - One second timestamp resolution.
  - Incremental backups require previous backup to be uncompressed.
- pgBackRest does not use rsync, tar or other typical backup tools:
  - Protocol supports local/remote operation.
  - Solves timestamp resolution issue.

# Multi-Process Backup & Restore

- Compression is usual bottleneck:
  - But most PostgreSQL backup solutions are single-process.
  - pgBackRest solves the problem with multi-processing.
  - 1TB/hr raw throughput even on a 1Gb/s link using multiple cores.

# Multi-Process Backup & Restore

- Compression is usual bottleneck:
  - But most PostgreSQL backup solutions are single-process.
  - pgBackRest solves the problem with multi-processing.
  - 1TB/hr raw throughput even on a 1Gb/s link using multiple cores.



# Multi-Process Backup & Restore

- Compression is usual bottleneck:
  - But most PostgreSQL backup solutions are single-process.
  - pgBackRest solves the problem with multi-processing.
  - 1TB/hr raw throughput even on a 1Gb/s link using multiple cores.

# Multi-Process Backup & Restore

- Compression is usual bottleneck:
  - But most PostgreSQL backup solutions are single-process.
  - pgBackRest solves the problem with multi-processing.
  - 1TB/hr raw throughput even on a 1Gb/s link using multiple cores.

# Local or Remote Operation

- Custom protocol allows backup, restore, and archive locally or remotely via SSH with minimal configuration.
- No direct access to PostgreSQL is required from the remote server which enhances security.

# Local or Remote Operation

- Custom protocol allows backup, restore, and archive locally or remotely via SSH with minimal configuration.
- No direct access to PostgreSQL is required from the remote server which enhances security.

# Full, Incremental, & Differential Backups

- Multiple backup types:
  - Full
  - Differential
  - Incremental
- pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe.

# Full, Incremental, & Differential Backups

- Multiple backup types:
  - Full
    - Differential
    - Incremental
  - pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe.

# Full, Incremental, & Differential Backups

- Multiple backup types:
  - Full
  - Differential
  - Incremental
- pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe.

# Full, Incremental, & Differential Backups

- Multiple backup types:
  - Full
  - Differential
  - Incremental
- pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe.



# Full, Incremental, & Differential Backups

- Multiple backup types:
  - Full
  - Differential
  - Incremental
- pgBackRest is not susceptible to the time resolution issues of rsync, making differential and incremental backups safe.

# Backup Rotation & Archive Expiration

- Retention Based on full or differential backups.
- WAL retention for all backups or configure number of recent backups.
- WAL required for consistency of backups always preserved.

# Backup Rotation & Archive Expiration

- Retention Based on full or differential backups.
- WAL retention for all backups or configure number of recent backups.
- WAL required for consistency of backups always preserved.

# Backup Rotation & Archive Expiration

- Retention Based on full or differential backups.
- WAL retention for all backups or configure number of recent backups.
- WAL required for consistency of backups always preserved.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.



# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
    - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Integrity

- Checksums are calculated for every file in the backup and rechecked during a restore.
- After a backup required WAL segments are checked in the repository.
- Simple backup format:
  - Backup directories have the same format as a PostgreSQL cluster.
  - Clusters can be brought up in place with snapshots if compression is disabled.
  - Advantageous for terabyte-scale databases.
- All operations utilize file and directory level fsync to ensure durability.

# Backup Resume

- An aborted backup can be resumed from the point where it stopped.
- Checksumming files on resume takes place on the backup server.
- Saves load on the master by not compressing and transmitting resumed files.

# Backup Resume

- An aborted backup can be resumed from the point where it stopped.
- Checksumming files on resume takes place on the backup server.
- Saves load on the master by not compressing and transmitting resumed files.

# Backup Resume

- An aborted backup can be resumed from the point where it stopped.
- Checksumming files on resume takes place on the backup server.
- Saves load on the master by not compressing and transmitting resumed files.

# Streaming Compression & Checksums

- Compression and checksum calculations are performed in stream.
- Compression is not done more than once.
- Lower compression is used when the destination is uncompressed to efficiently utilize CPU and network bandwidth.

# Streaming Compression & Checksums

- Compression and checksum calculations are performed in stream.
- Compression is not done more than once.
- Lower compression is used when the destination is uncompressed to efficiently utilize CPU and network bandwidth.



# Streaming Compression & Checksums

- Compression and checksum calculations are performed in stream.
- Compression is not done more than once.
- Lower compression is used when the destination is uncompressed to efficiently utilize CPU and network bandwidth.

# Delta Restore

- Backup manifest contains checksum and size for every file.
- On delta restore all files not present in the backup or with a different size are removed from PGDATA.
- The remaining files are checksummed and only files with a checksum mismatch are restored.
- Multi-processing can lead to dramatic reductions in restore time and network utilization.

# Delta Restore

- Backup manifest contains checksum and size for every file.
- On delta restore all files not present in the backup or with a different size are removed from PGDATA.
- The remaining files are checksummed and only files with a checksum mismatch are restored.
- Multi-processing can lead to dramatic reductions in restore time and network utilization.

# Delta Restore

- Backup manifest contains checksum and size for every file.
- On delta restore all files not present in the backup or with a different size are removed from PGDATA.
- The remaining files are checksummed and only files with a checksum mismatch are restored.
- Multi-processing can lead to dramatic reductions in restore time and network utilization.

# Delta Restore

- Backup manifest contains checksum and size for every file.
- On delta restore all files not present in the backup or with a different size are removed from PGDATA.
- The remaining files are checksummed and only files with a checksum mismatch are restored.
- Multi-processing can lead to dramatic reductions in restore time and network utilization.

# Advanced Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous archiving allows compression and transfer to be offloaded to another process which maintains a continuous connection to the remote server, improving throughput significantly.
  - Critical feature for databases with extremely high write volume.

# Advanced Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous archiving allows compression and transfer to be offloaded to another process which maintains a continuous connection to the remote server, improving throughput significantly.
  - Critical feature for databases with extremely high write volume.

# Advanced Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous archiving allows compression and transfer to be offloaded to another process which maintains a continuous connection to the remote server, improving throughput significantly.
  - Critical feature for databases with extremely high write volume.



# Advanced Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous archiving allows compression and transfer to be offloaded to another process which maintains a continuous connection to the remote server, improving throughput significantly.
  - Critical feature for databases with extremely high write volume.

# Advanced Archiving

- Dedicated commands are included for both pushing WAL to the archive and retrieving WAL from the archive.
- Push command automatically detects WAL segments that are pushed multiple times and de-duplicates when the segment is identical, otherwise an error is raised.
- Push and get commands both ensure that the database and repository match by comparing PostgreSQL versions and system identifiers to prevent misconfiguration.
- Asynchronous archiving allows compression and transfer to be offloaded to another process which maintains a continuous connection to the remote server, improving throughput significantly.
  - Critical feature for databases with extremely high write volume.

# Tablespace & Link Support

- Tablespaces are fully supported and on restore tablespaces can be remapped to any location.
- Remap all tablespaces to one location with a single command which is useful for development restores.
- File and directory links are supported for any file or directory in the PostgreSQL cluster.
- Restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory

# Tablespace & Link Support

- Tablespaces are fully supported and on restore tablespaces can be remapped to any location.
- Remap all tablespaces to one location with a single command which is useful for development restores.
- File and directory links are supported for any file or directory in the PostgreSQL cluster.
- Restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory

# Tablespace & Link Support

- Tablespaces are fully supported and on restore tablespaces can be remapped to any location.
- Remap all tablespaces to one location with a single command which is useful for development restores.
- File and directory links are supported for any file or directory in the PostgreSQL cluster.
- Restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory

# Tablespace & Link Support

- Tablespaces are fully supported and on restore tablespaces can be remapped to any location.
- Remap all tablespaces to one location with a single command which is useful for development restores.
- File and directory links are supported for any file or directory in the PostgreSQL cluster.
- Restore all links to their original locations, remap some or all links, or restore some or all links as normal files or directories within the cluster directory

# Selective Restore

- Restore only specified databases out of a cluster backup.
- Other files are restored as sparse, zeroed files the save space.
- All WAL must be replayed.
- Cannot connect to non-restored databases, can only drop them.

# Selective Restore

- Restore only specified databases out of a cluster backup.
- Other files are restored as sparse, zeroed files the save space.
- All WAL must be replayed.
- Cannot connect to non-restored databases, can only drop them.



# Selective Restore

- Restore only specified databases out of a cluster backup.
- Other files are restored as sparse, zeroed files the save space.
- All WAL must be replayed.
- Cannot connect to non-restored databases, can only drop them.

# Selective Restore

- Restore only specified databases out of a cluster backup.
- Other files are restored as sparse, zeroed files the save space.
- All WAL must be replayed.
- Cannot connect to non-restored databases, can only drop them.

# Selective Restore

- Restore only specified databases out of a cluster backup.
- Other files are restored as sparse, zeroed files to save space.
- All WAL must be replayed.
- Cannot connect to non-restored databases, can only drop them.

# Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

# Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

# Backup from Standby

- Backup is started on master.
- Backup starts when replay location on standby reaches start backup location.
- Reduces load on master because replicated files are copied from the standby.

# Compatibility with PostgreSQL $\geq$ 8.3

- Support for versions down to 8.3, since older versions of PostgreSQL are still regularly utilized.

# Performance

Parameters	pgBackRest	rsync
processes: 1 network compression: l3 destination compression: none	141 Seconds	124 Seconds (.13X Faster)
processes: 2 network compression: l3 destination compression: none	84 Seconds (1.48X Faster)	N/A
processes: 1 network compression: l6 destination compression: l6	334 Seconds (1.52X Faster)	510 Seconds
processes: 2 network compression: l6 destination compression: l6	174 Seconds (2.93X Faster)	N/A



# Demonstration

Live Demo — this should be fun!

# Questions?

website: <http://www.pgbackrest.org>

email: [david@pgbackrest.org](mailto:david@pgbackrest.org)

email: [david@crunchydata.com](mailto:david@crunchydata.com)

releases: <https://github.com/pgbackrest/pgbackrest/releases>

slides & demo: <https://github.com/dwsteele/conference/releases>