PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Introduction

PERFORCE
Version everything.

- Introductions
- Class Schedule
- GUI vs. CLI
- P4Admin Demonstrations
- About the Exercises

Course Contents

PERFORCE
Version everything.

- Advanced Maintenance
- Offline Checkpoints
- Broker
- Replication – Introduction
- Replication – HA/DR, Build Farms, Forwarding Replicas
- Fully Distributed
- Security
- Advanced Tools
- Scripting

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Advanced Maintenance

Topics

PERFORCE
Version everything.

- Recover a Stored Spec Revision
- Lazy Copies
- Archive/Restore

Spec Depot

PERFORCE
Version everything.

- **Goal**
  - Recover specs such as clients and protection table
  - Keep history of changes to specs
  - Identify user who changed a spec

- **Implementation**
  - Separate spec depot automatically maintained by Perforce Server
  - Specs are stored as form files, which can be printed or synced
    - Grouped into directories by type, such as *client* or *label*

Spec Depot Usage

PERFORCE
Version everything.

- Spec depot stores specs like clients and protection table (not change)
- Tracing of changes by a user

```
p4 print -q //spec/label/lastbuild.p4s#1
# The form data below was edited by bruno
```

- Optional: controlling which specs are versioned

```
p4 depot specs
SpecMap:
    //specs/...
    -//specs/client/build_ws_*
```

Recovering a Stored Spec Revision

PERFORCE
Version everything.

- List revisions in the spec depot

```
p4 filelog //specs/client/bruno_ws.p4s
... #4 default change edit on 2014/11/01
... #3 default change edit on 2014/10/17
... #2 default change edit on 2014/07/01
... #1 default change add on 2013/11/20
```

- Display content of revisions

```
p4 print -a //specs/client/bruno_ws.p4s
```

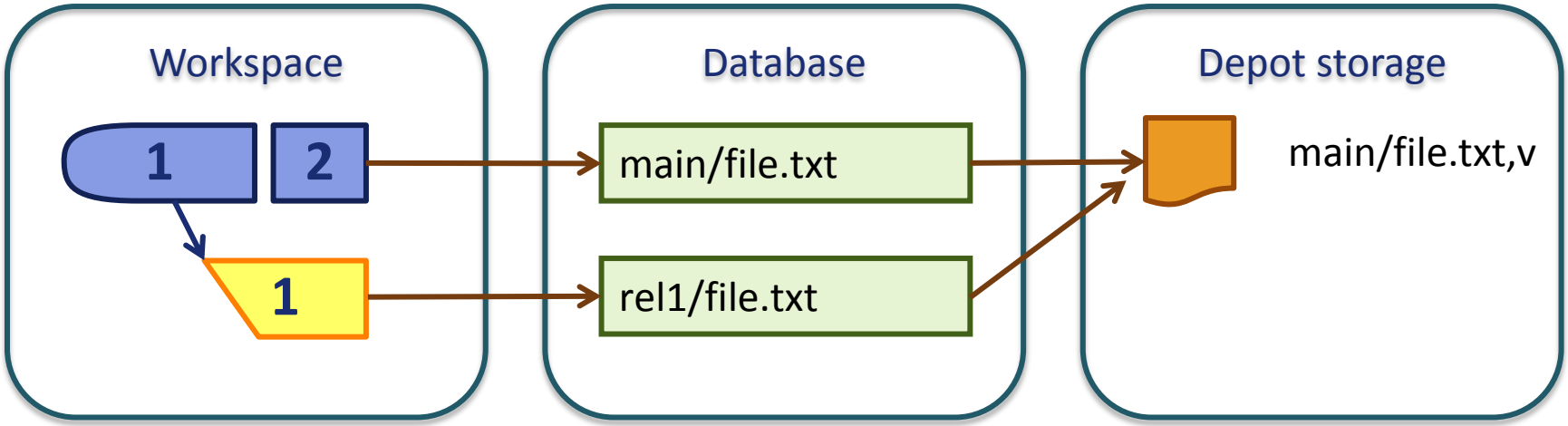
- Replace spec with earlier version

```
p4 print -q //specs/client/bruno_ws.p4s#3 | p4 client -i
```

Branching and Lazy Copies

PERFORCE
Version everything.

- Files branched or copied only create metadata entry in the database
  - Retain reference to original file location ➔ lazy copy



Lazy Copies and Snap

PERFORCE
Version everything.

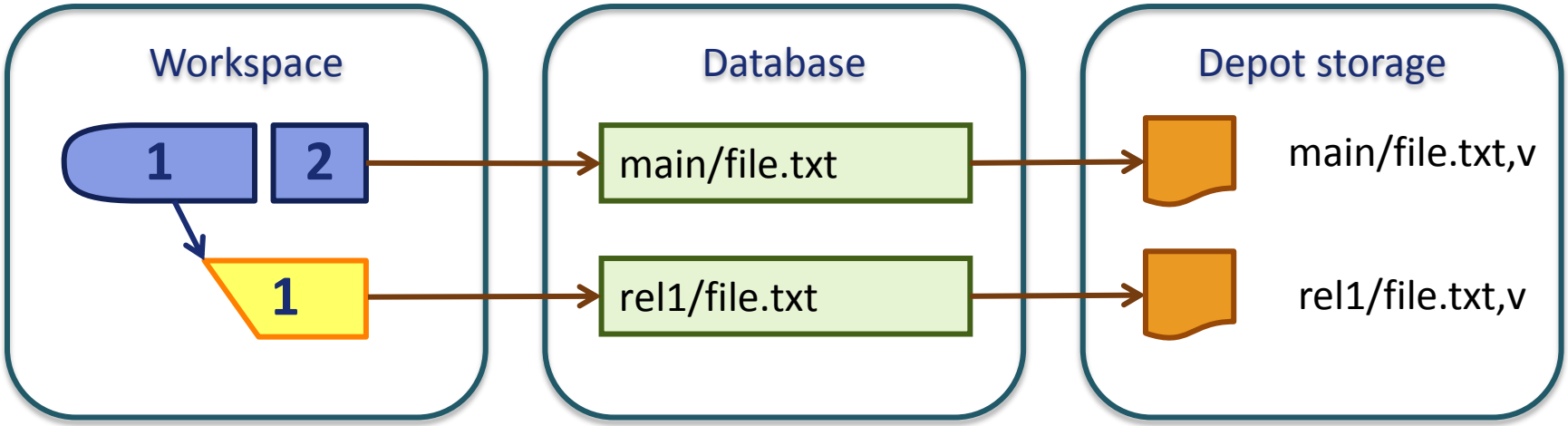
```
p4 fstat -Oc //depot/Jam/REL2.0/src/jam.c
...
... lbrFile //depot/Jam/MAIN/src/jam.c
... lbrRev 1.30
... lbrType text
... lbrIsLazy 1
```

(undocumented)

```
p4 snap //depot/Jam/REL2.0/src/jam.c
//depot/Jam/REL2.0/src/jam.c#1 -
        copy from //depot/Jam/MAIN/src/jam.c 1.30
```

After Snap

- Files in the depot storage are duplicated
- Useful when cleaning up depots with *obliterate*



Archiving and Restoring

PERFORCE
Version everything.

- Goal:
  - Free up space in active depots
  - Speed up backup and verify
  - Preserve history
  - Simple restore

- Implementation:
  - Separate archive depots (typically located on cheap storage)
  - Files can be archived and restored at individual revisions

Archiving and Restoring

PERFORCE
Version everything.

- Files *not branched* can be archived
  - Requires at least one depot of type *archive*
  - Preserves history

```
p4 archive -D archives //assets/...
```

- To archive files stored in delta format, use the -t option.

```
p4 archive -D archives -t //assets/text/readme.txt#9,9
```

- Restore files as needed

```
p4 restore -D archives  //assets/images/myimage.jpg#3
```

Archiving – Listing and Purging

PERFORCE
Version everything.

- Files in original depot are marked as *archive*

```
p4 files //assets/...
//assets/images/myimage.gif#1 - archive change 865 (ubinary)
...
```

- List files in archive depot

```
p4 files -A //archives/...
//archives/assets/images/myimage.gif#1
...
```

- Purge unneeded archived files (cannot be undone)

```
p4 archive -D archives -p //assets/...@2012/01/01
```

Exercises

Lab Set E1: Advanced Maintenance

New commands in this chapter (samples):

- `p4 archive`
- `p4 restore`
- `p4 snap (undoc)`

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Offline Checkpoints

Topics

PERFORCE
Version everything.

- Offline Checkpoints
  - Usage
  - Upgrades
  - Switch offline_db/root

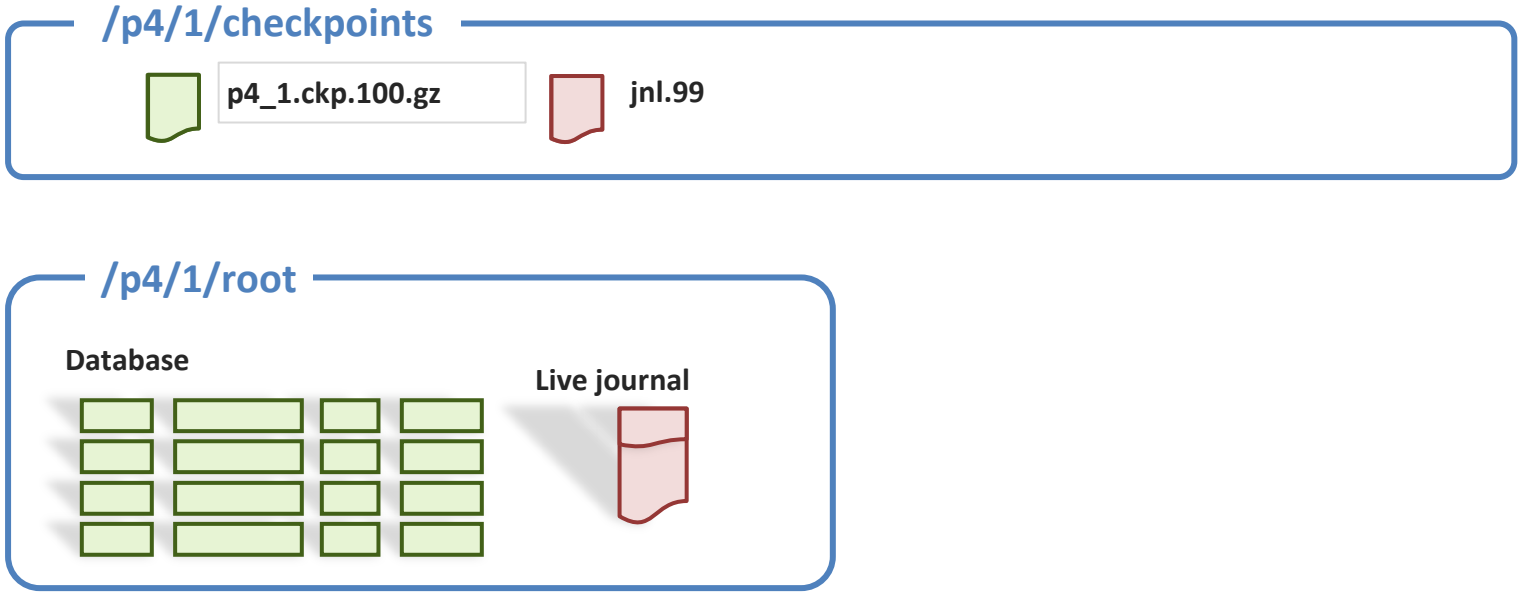
Offline Checkpoint

PERFORCE
Version everything.

- **Goal**
  - Checkpoint without any downtime
  - Easy and fast recovery
  - Optional: regular database restoration
    - Restored databases are smaller than original, but contain equivalent data (Removes empty data pages and rebalances the b-tree indexes)
- **Implementation**
  - Separate offline database created from checkpoint
  - Regular updates through rotated journal
  - Offline database dumped into checkpoint

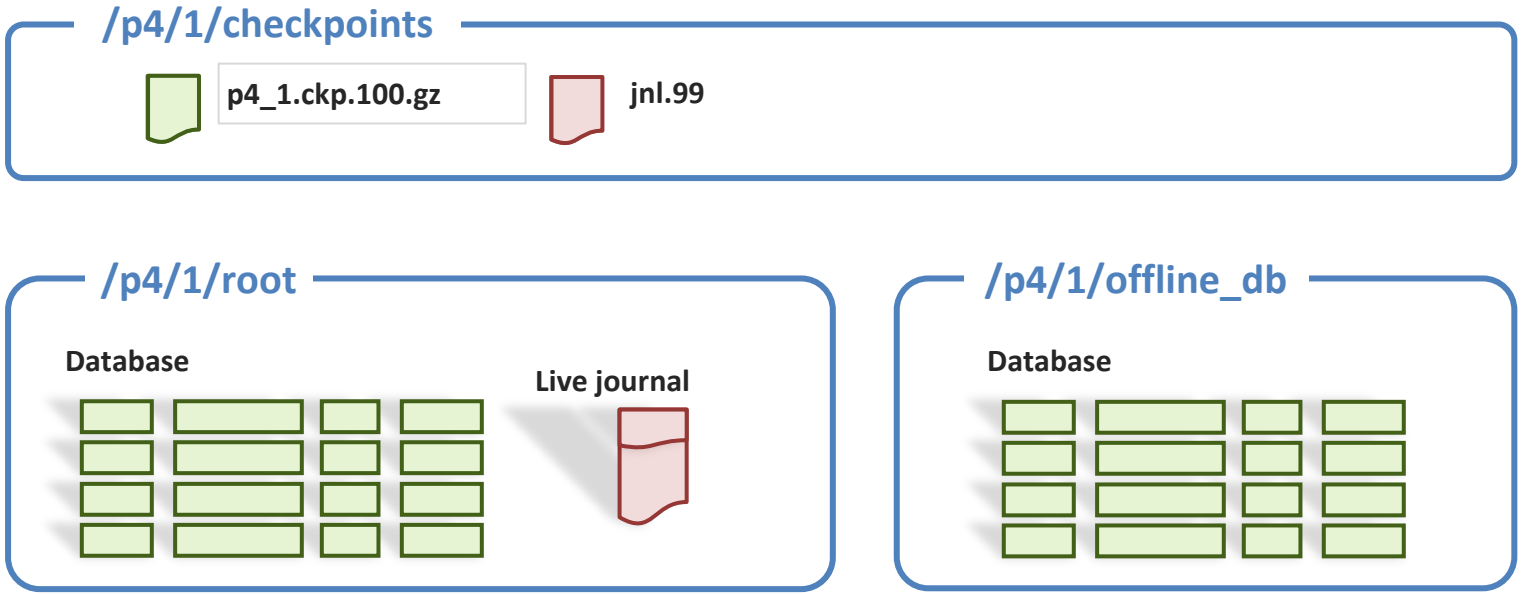
Prep Offline Checkpoint – Create Seed

```
p4d -r /p4/1/root -jc -Z /p4/1/checkpoints/p4_1
```



Prep Offline Checkpoint – Apply Seed

```
p4d -r /p4/1/offline_db -jr -z /p4/1/checkpoints/p4_1.ckp.100.gz
```



Offline Checkpoint

PERFORCE
Version everything.

- Nightly:
  - Truncate journal on live database

```
p4d -r /p4/1/root -J /p4/1/logs/journal -jj /p4/1/checkpoints/p4_1
```

  - Restore journal to offline directory

```
p4d -r /p4/1/offline_db -jr /p4/1/checkpoints/p4_1.jnl.100
```

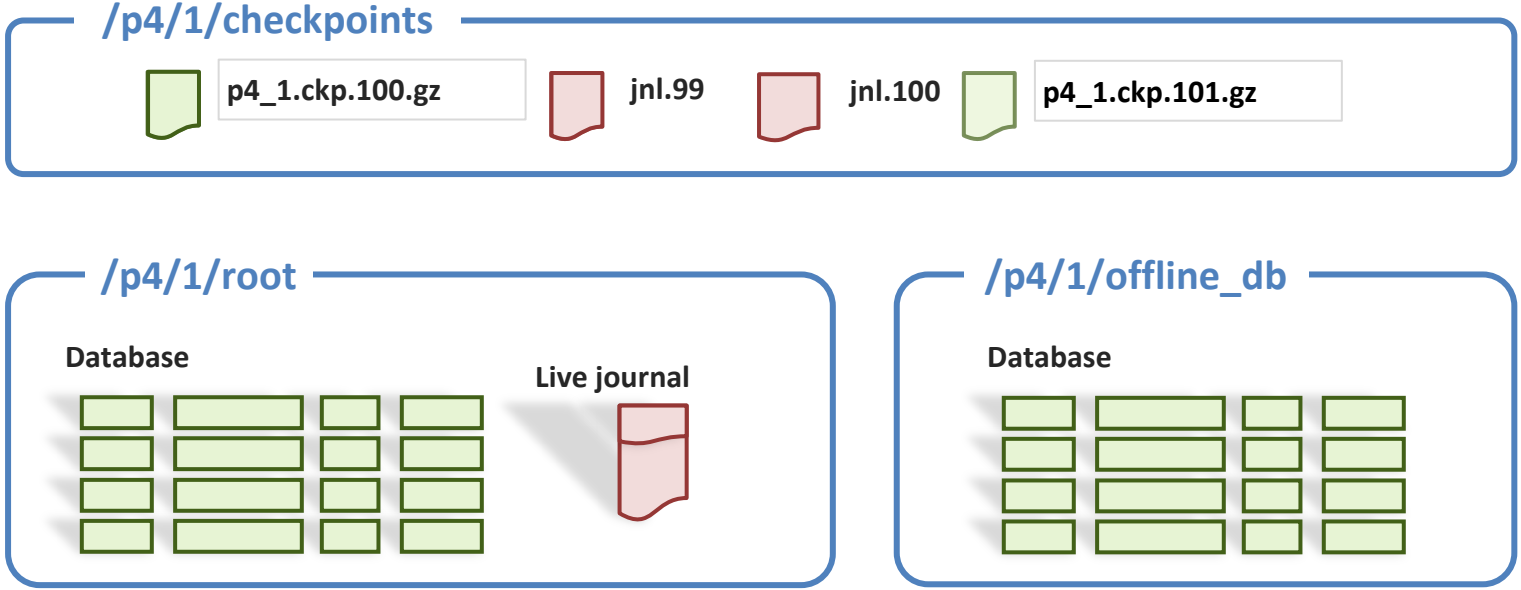
  - Dump the offline database to make a new checkpoint

```
p4d -r /p4/1/offline_db -jd -z /p4/1/checkpoints/p4_1.ckp.101.gz
```

Offline Checkpoint

PERFORCE
Version everything.

Truncate journal `-jj` › Restore journal `-jr` › Dump checkpoint `-jd`



Recreate Offline Database

PERFORCE
Version everything.

- Recreate the offline database from the new checkpoint

```
rm -f /p4/1/offline_db/db.*
p4d -r /p4/1/offline_db -jr -z /p4/1/checkpoints/p4_1.ckp.101.gz
```

Switch Offline Database/Root

PERFORCE
Version everything.

- Stop the production server
- Rotate the journal
- Replay the journal to the offline_db
- Move /p4/1/root/db.* /p4/1/root/save/
- Move /p4/1/offline_db/db.* /p4/1/root/
- Restart the master server
- Dump a checkpoint from /p4/1/root/save
- Recreate the offline database from the new checkpoint

Exercises

- Lab Set E2: Offline Checkpoints

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Broker

P4Broker

PERFORCE
Version everything.

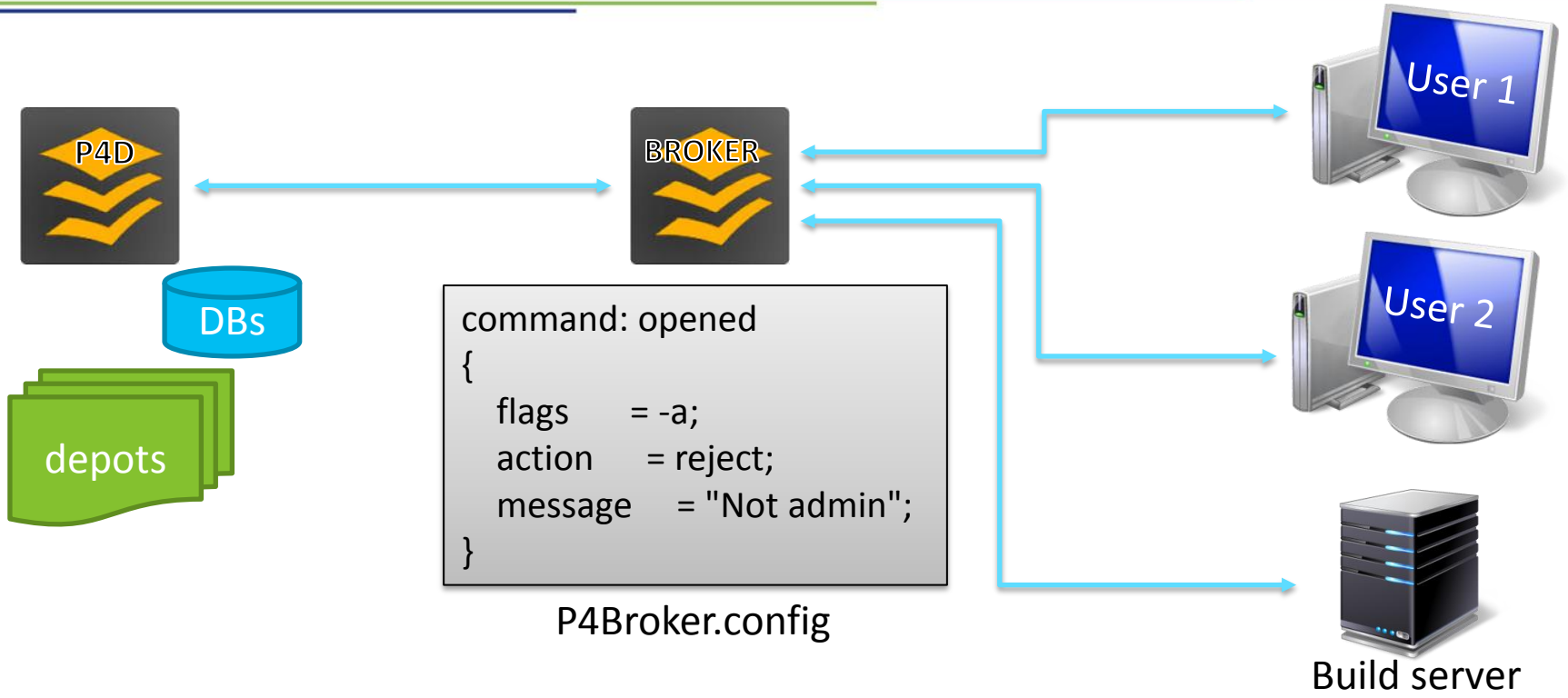
- Intercepts all incoming Perforce commands
- Command handling support:
  - Redirection
  - Blocking
  - Rewriting (undocumented)
- Great for notifying users when the server is down for maintenance.
- Sometimes used as part of HA/DR strategies to avoid DNS change delay.

P4Broker Use Cases

PERFORCE
Version everything.

- Policy Customizations
  - different capabilities than triggers
- Traffic Redirection for Load Distribution
  - not “load balancing”
- Traffic Redirection for execution of automated failover operations
  - advanced/custom usage

Perforce Broker



Redirection

PERFORCE
Version everything.

- Selective – The default setting
  - Redirection allowed, but after the first command in a session hits the default server, all others in the same session use the default server and are not redirected.

- Pedantic – All redirected commands are redirected
  - Can cause the GUI to not update the icons correctly.

Read Only Commands

PERFORCE
Version everything.

- Redirect read only commands to replica servers to offload the master server
- P4Broker can do load balancing
  - Use the name “random” for the target server in a redirected command handler.
- http://kb.perforce.com/article/1354 -- Using P4Broker With Replica Servers
- (This particular use case can now be handled with a forwarding replica.)

Filter Scripts

PERFORCE
Version everything.

- When the action for a command handler is “filter”:
  - The broker executes the program or script and performs the action returned by the program.
- The broker invokes the filter program and passes in all the information about the command via stdin.
  - Your filter program must read this data from stdin before performing any additional processing.
- The filter program responds on stdout with one of the following:
  - action: PASS/REJECT/REDIRECT/RESPOND
  - message: Some message for the user

Mechanics: P4Broker Setup

PERFORCE
Version everything.

- Define an operating server.
- Generate a preliminary broker configuration file.
- Adjust the broker configuration to your needs.
- Set broker config file location.
- Initiate as a Windows service or Unix/Linux daemon.
- Documentation:
  - Latest Release Broker Notes
  - Distributing Perforce Manual

Exercises

PERFORCE
Version everything.

- Lab Set E3: P4Broker

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Replication - Introduction

Why Replication?

PERFORCE
Version everything.

- Disaster Recovery
  - Possibly read-only
- Offloading intense server traffic
  - Reports
  - Builds
- Forwarding Replica (aka Smart Proxy)
- Edge / Commit server architecture (distributed working)

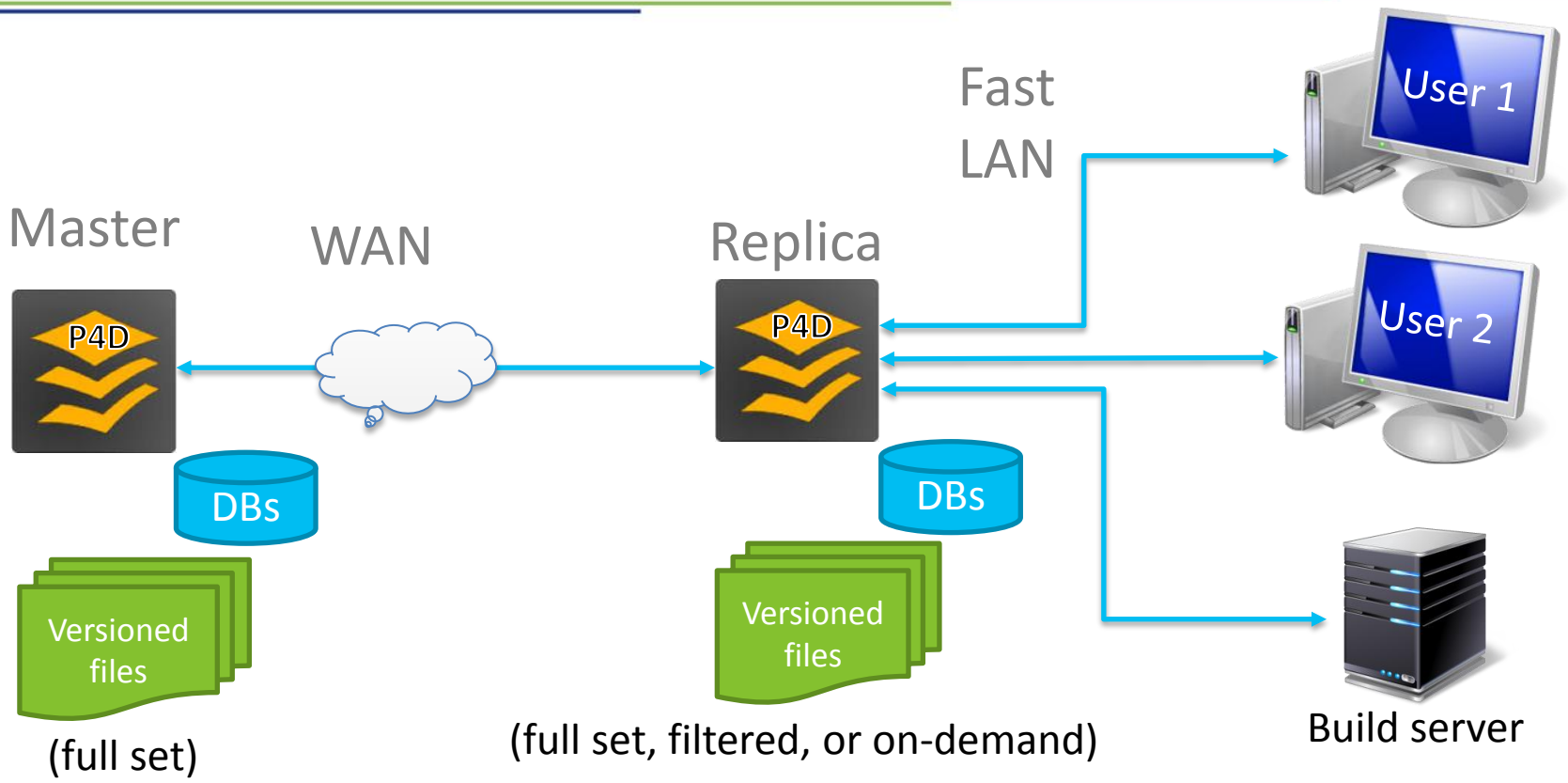
Perforce Replication - Implementation

PERFORCE
Version everything.

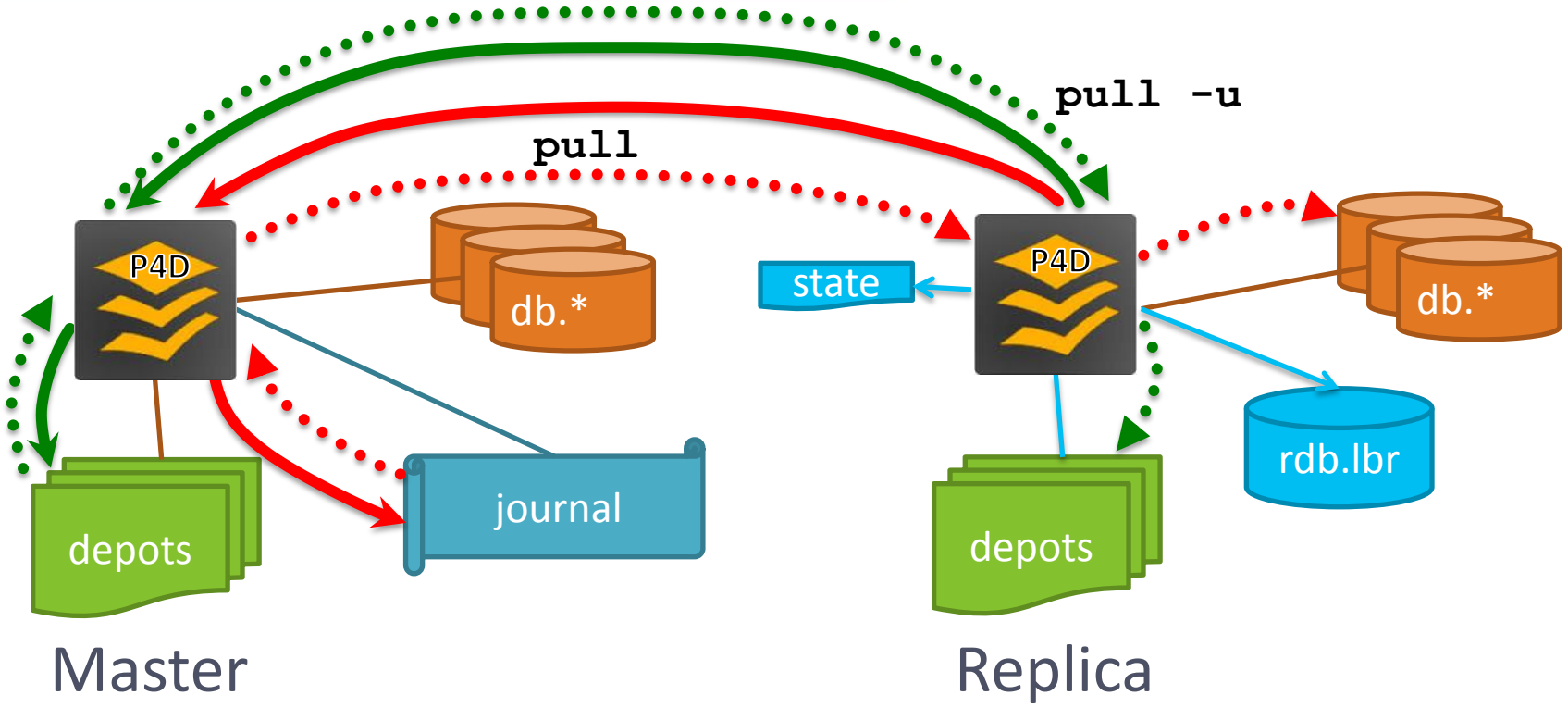
- Server-to-Server replication
  - Asynchronous based on journal file
  - Supports both Metadata-only and Full Replication
  - No need for external scripts, complete solution

- Replicas must initially be seeded with a checkpoint (metadata).
  - Versioned files are required for full replication
    - Can be copied after setup using Perforce replication

Replication Architecture – General



Replication Architecture – Detailed



Naming Servers

PERFORCE
Version everything.

- All Perforce servers should be named and have server specifications
- Server names…
  - Are used in replication and failover and other scenarios
  - Define server capabilities
  - Determine which configurables apply to a server
  - Enforce security
    - Can require special service accounts for access by remote servers

Naming Servers

PERFORCE
Version everything.

- `p4 serverid [serverID]`
- `p4d -xD [serverID]`
  - Sets/retrieves server.id file in server's root directory
- Tells server which configurables apply

- `P4NAME`
  - Environment variable
  - Overrides server.id file

Configurables and Named Servers

PERFORCE
Version everything.

- `p4 configure set replica#P4TARGET=192.168.1.1#1666`
- `p4 configure show`
  - Shows running configuration of queried server
- `p4 configure show allservers`
  - Shows stored configurables for all servers
- `p4 configure show replica`
  - Shows configurables stored in queried server for server named ‘replica’

Configurables and Named Servers

PERFORCE
Version everything.

```
> p4 configure show
P4ROOT=. (-r)
P4PORT=9876 (-p)
P4JOURNAL=journal (default)
threadingmode=true (-d)
p4zk.log.file=p4zk.log (default)
auth.default.method=perforce (default)
>
> p4 configure show replica
replica: P4TARGET = 127.0.0.1:9876
replica: P4TICKETS = /path/to/replica/.p4tickets
replica: db.replication = readonly
replica: lbr.replication = readonly
replica: startup.1 = pull -i 1
replica: startup.2 = pull -u -i 1
```

Server Specifications

- `p4 server servername`
  - Creates or updates information about a server
  - Specifies information about a server such as its type

Type	Definition
standard	Standard Server
replica	Replica Server
broker	P4Broker
proxy	Perforce Proxy
forwarding-replica	Smart Proxy
build-server	Build Server
P4AUTH	Authentication Server
P4CHANGE	Change Server

p4 pull

PERFORCE
Version everything.

- Runs as a background task inside the replica

Command	Effect
`p4 pull`	Retrieve missing journal entries, then terminate
`p4 pull -i <N>`	Continuously pull every <N> seconds
`p4 pull -u`	Retrieve missing file revisions, then terminate
`p4 pull -u -i <N>`	Continuously pull file revisions
`p4 pull -l`	List missing file revisions or errors
`p4 pull -l [-j | -s]`	Replica reporting

- `p4 pull -lj` Shows metadata replication status
- `p4 pull -ls` Shows content transfer status

How does ‘p4 pull’ keep track?

PERFORCE
Version everything.

- **state** file
  - Text file normally located in the replica P4ROOT directory
  - `journal#/offset`
  - Allows replication to be interrupted
  - Master server can rotate journal file
    - Configure journalPrefix if master uses journal prefix for checkpoints

- **rdb.lbr** database
  - Binary file located in the replica P4ROOT directory
  - Contains information on missing archive revisions

Journal rotation and Prefix

PERFORCE
Version everything.

- **Master**
  - `p4 admin checkpoint/journal [-Z] prefix`
  - Do not use –z, use –Z (uppercase)
    - Compresses checkpoint but not rotated journal file
  - If you use a prefix, use the same prefix for ‘p4 pull’
    - Use ‘journalPrefix’ configurable (or pull -J)
- **Replica**
  - `p4 pull [-J prefix] [-i n]`
  - Journal will be rotated in sync with the master

journalPrefix

PERFORCE
Version everything.

- Specify journalPrefix on the master to
  - Simplify checkpoint and journal rotation
  - Avoid having to specify ‘p4 pull –J prefix’ in the replica(s)

- Specify journalPrefix in the replica to
  - Automatically rotate journal to correct location when master rotates
  - Help to prevent replica running out of disk space
  - Without journalPrefix, replica will rotate journal in P4ROOT

```
p4 configure set replica#journalPrefix=/replica/checkpoints/repl_1
```

Configuration

PERFORCE
Version everything.

- ‘p4 pull’ is designed to be a background process
  - Started from the replica server
  - One process for retrieving metadata
  - Several additional processes to retrieve archive data

- Use ‘p4 configure set’ for named servers

```
p4 configure set P4NAME#variable=value
p4 configure set repl_1#P4TARGET=192.168.1.1:1666
```

Prepare in the Master

PERFORCE
Version everything.

```
> p4 configure set monitor=1
For server 'any', configuration variable 'monitor' set to '1'

> p4 configure set Master#net.tcpsize=512k
For server 'Master', configuration variable 'net.tcpsize' set to '512k'
> p4 configure set Repl_1#P4TARGET=master:1666
For server 'Repl_1', configuration variable 'P4TARGET' set to 'master:1666'
```

Prepare in the Master

server.id=Master

checkpoint

server.id=Repl_1

restore

`p4 configure show allservers`

```
Any: monitor=1
Master: net.tcpsize=512k
Master: lbr.bufsize=64k
Repl_1: P4TARGET=master:1666
Repl_1: serviceUser=service
Repl_1: db.replication=readonly
Repl_1: lbr.replication=readonly
Repl_1: startup.1=pull -i 0
Repl_1: startup.2=pull -i 1 -u
```

`p4 configure show allservers`

```
Any: monitor=1
Master: net.tcpsize=512k
Master: lbr.bufsize=64k
Repl_1: P4TARGET=master:1666
Repl_1: serviceUser=service
Repl_1: db.replication=readonly
Repl_1: lbr.replication=readonly
Repl_1: startup.1=pull -i 0
Repl_1: startup.2=pull -i 1 -u
```

server.id determines which configuration is active

Server environment settings

PERFORCE
Version everything.

- Command line flags
- Configure set/cset
- Environment variables
- (On Windows) registry variables

Configuration parameters

PERFORCE
Version everything.

Parameter	Sample Values
P4TARGET	master:1666
db.replication	readonly
lbr.replication	readonly
rpl.forward.all	1
serviceUser	replica_1_svc
startup.1	pull –i 0
startup.2	pull –u –i 1
startup.3	pull –u –i 1

Service user

- Replication requires user of type `service`.
- The service user requires ‘super’ access.
- Add the user to a group (e.g. service.g) group with unlimited timeout.

- On replica machine login as the service user before starting replication
  - Define P4TICKETS location for the replica

```
p4 set -s P4TICKETS=c:\p4\p4tickets.txt
p4 -u p4admin login service_user
```

Active Replication Monitoring

PERFORCE
Version everything.

- `p4 pull -l [-j|-s]`
  - Reports pending transfers
- `p4 verify [-t]`
  - Option -t schedules content transfer of missing/damaged revision
- `p4 journaldbchecksums`
  - Run on master, check log on replica

Simple replica setup - master

PERFORCE
Version everything.

- Set up the replica environment on the **master** server
- Create a replica service user:
  - `p4 user -f replica_svc_user`
  - Add **`Type: service`** to the spec and save it
  - `p4 passwd replica_svc_user`
- Create a server specification:
  - `p4 server rep_1`
  - Add **`Services: forwarding-replica`** to the spec and save it

Simple replica setup - master

PERFORCE
Version everything.

- Set variables for the replica in the master:

```
p4 configure set server=3
p4 configure set rep_1#P4TARGET=192.168.1.1:1666
p4 configure set rep_1#P4TICKETS=/path/to/.p4tickets
p4 configure set "rep_1#startup.1=pull -i 1"
p4 configure set "rep_1#startup.2=pull -u -i 1"
p4 configure set rep_1#db.replication=readonly
P4 configure set rep_1#lbr.replication=readonly
```

Simple replica setup - master

PERFORCE
Version everything.

- Verify settings on master:

```
> p4 configure show rep_1
rep_1: P4TARGET = 192.168.1.1:1666
rep_1: P4TICKETS = /path/to/replica/.p4tickets
rep_1: db.replication = readonly
rep_1: lbr.replication = readonly
rep_1: startup.1 = pull -i 1
rep_1: startup.2 = pull -u -i 1
```

- All okay? Take a checkpoint of the master:

```
p4 admin checkpoint -Z
```

Simple replica setup - replica

PERFORCE
Version everything.

- Copy the checkpoint to the replica and restore
- Create the server.id file on the replica:

```
p4d -r . -xD rep_1
Perforce server info:
        Server ID: rep_1
```

- Log into the master from replica

```
p4 -p 192.168.1.1:1666 -u replica_svc_user login
```

  - Ensure user's P4TICKETS file points to the same location as the replica's setting
- Start the replica

Simple replica setup - replica

PERFORCE
Version everything.

- Replication is working:

```
> p4 pull -lj
Current replica journal state is:       Journal 2,      Sequence 683.
Current master journal state is:        Journal 2,      Sequence 683.
The statefile was last modified at:     2014/10/30 14:27:56.
The replica server time is currently:   2014/10/30 14:28:38 -0700 PDT
>
> p4 -p 192.168.1.1:1666 journaldbchecksums
Perforce server info:
      Table db.config checksums match. 2I li014/10/30 14:33:41 version 1:
expected
Perforce server info:
      Table db.counters checksums match. 2014/10/30 14:33:41 version 1: expecte
Perforce server info:
      Table db.nameval checksums empty. 2014/10/30 14:33:41 version 1: expected
```

Replication *live*

PERFORCE
Version everything.

- Replication really is easy to configure
  - Setup and install of forwarding replica

Exercises

Lab Set E4: Replication

New commands in this chapter:

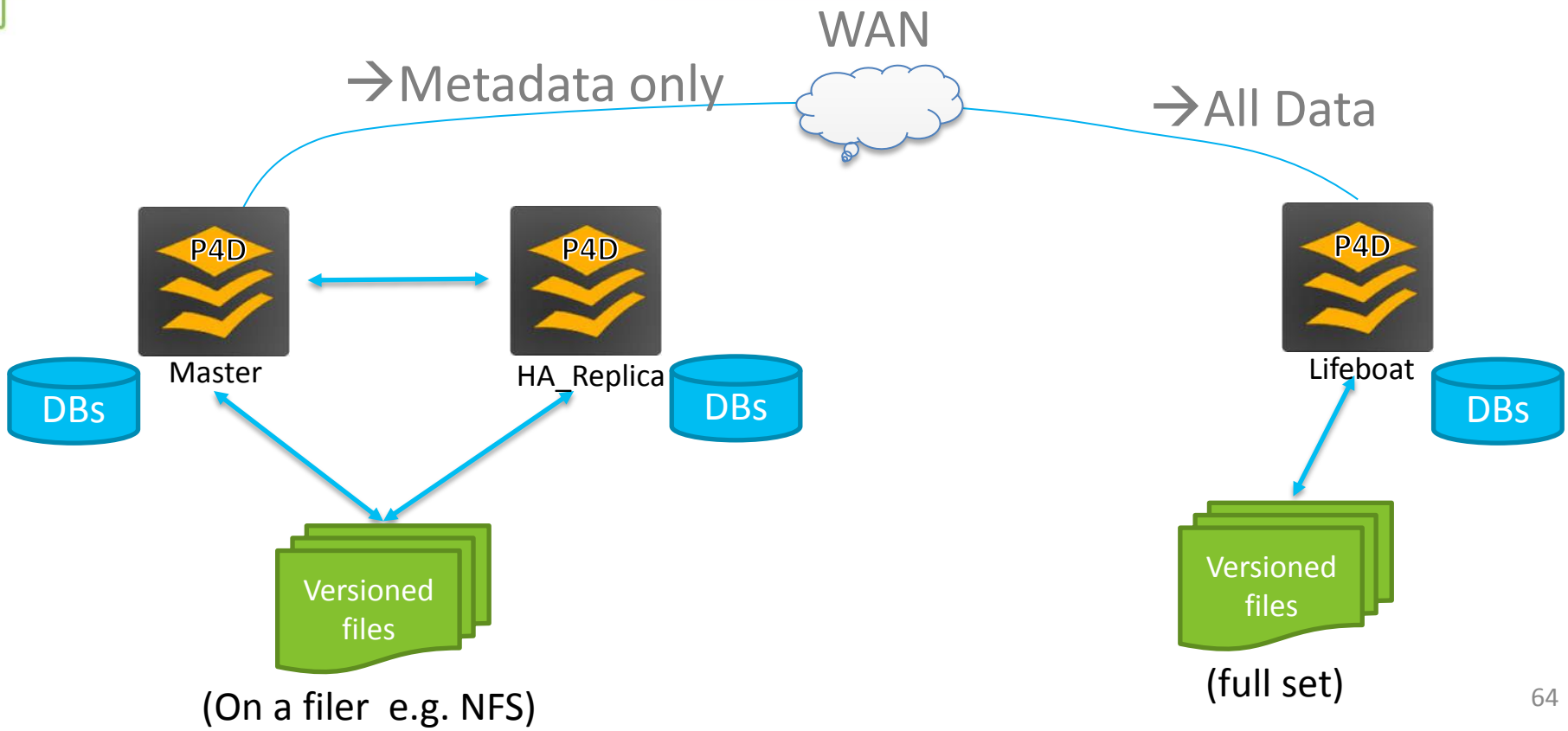
- `p4 configure set P4NAME#var=val`
- `p4 configure show allservers`
- `p4 pull`
- `p4 pull -l [-j | -s]`
- `p4 journaldbchecksums`
- `p4 verify -t`
- `p4d -xD`
- `p4 server`

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

More Replication Options

Replicas for HA and DR



Prepare in the Master

PERFORCE
Version everything.

```
p4 server Replica1

ServerID: Replica1
Name: Replica1
Type: server
Services: forwarding-replica
```

```
p4 configure set Replica1#db.replication=readonly
p4 configure set Replica1#lbr.replication=readonly
```

Equivalent value set via ‘p4 server’ specification:

```
p4 configure set Replica1#rpl.forward.all=1
```

Replica filtering

- To exclude entire tables from a replica:

```
p4 pull -T db.have,db.client
```

- Detailed Filtering:

```
p4 server Replica1
ServerID: Replica1
:
ClientDataFilter:
  -//site2-ws-*
ArchiveDataFilter:
  //....c
  -//....mp4
```

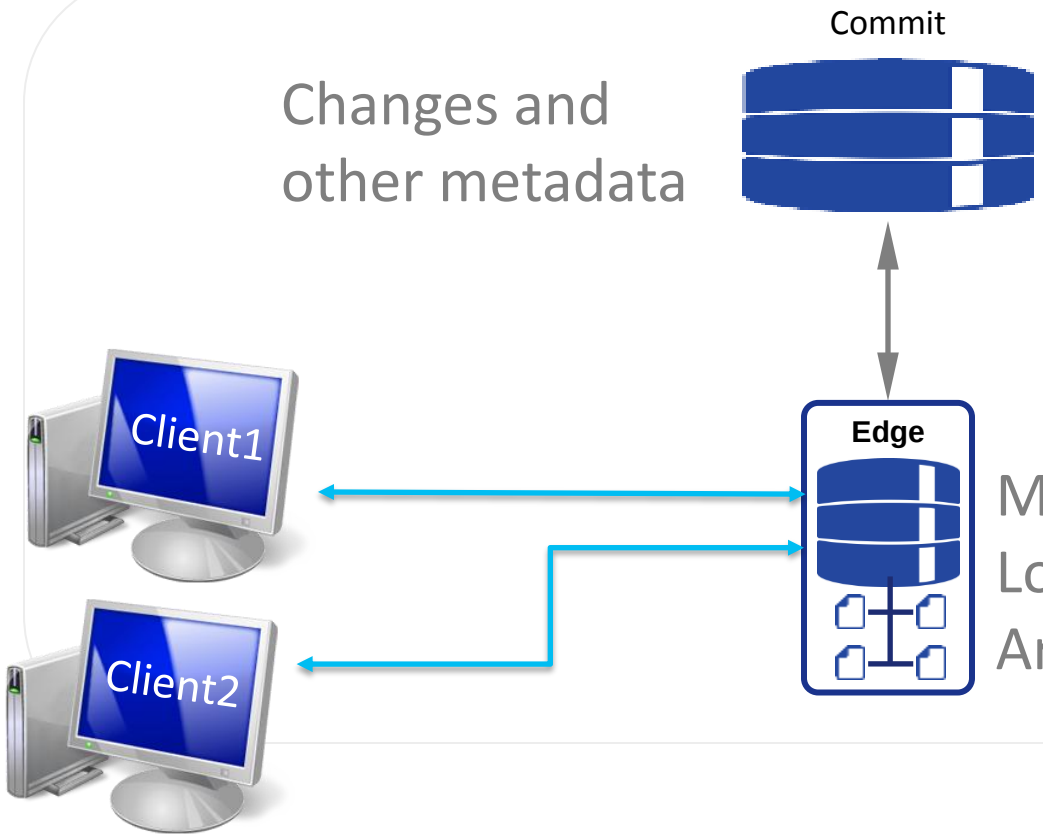
```
p4 configure set
"Replica1#startup.1=pull -i
30 -P Replica1"
```

PERFORCE
Version everything.

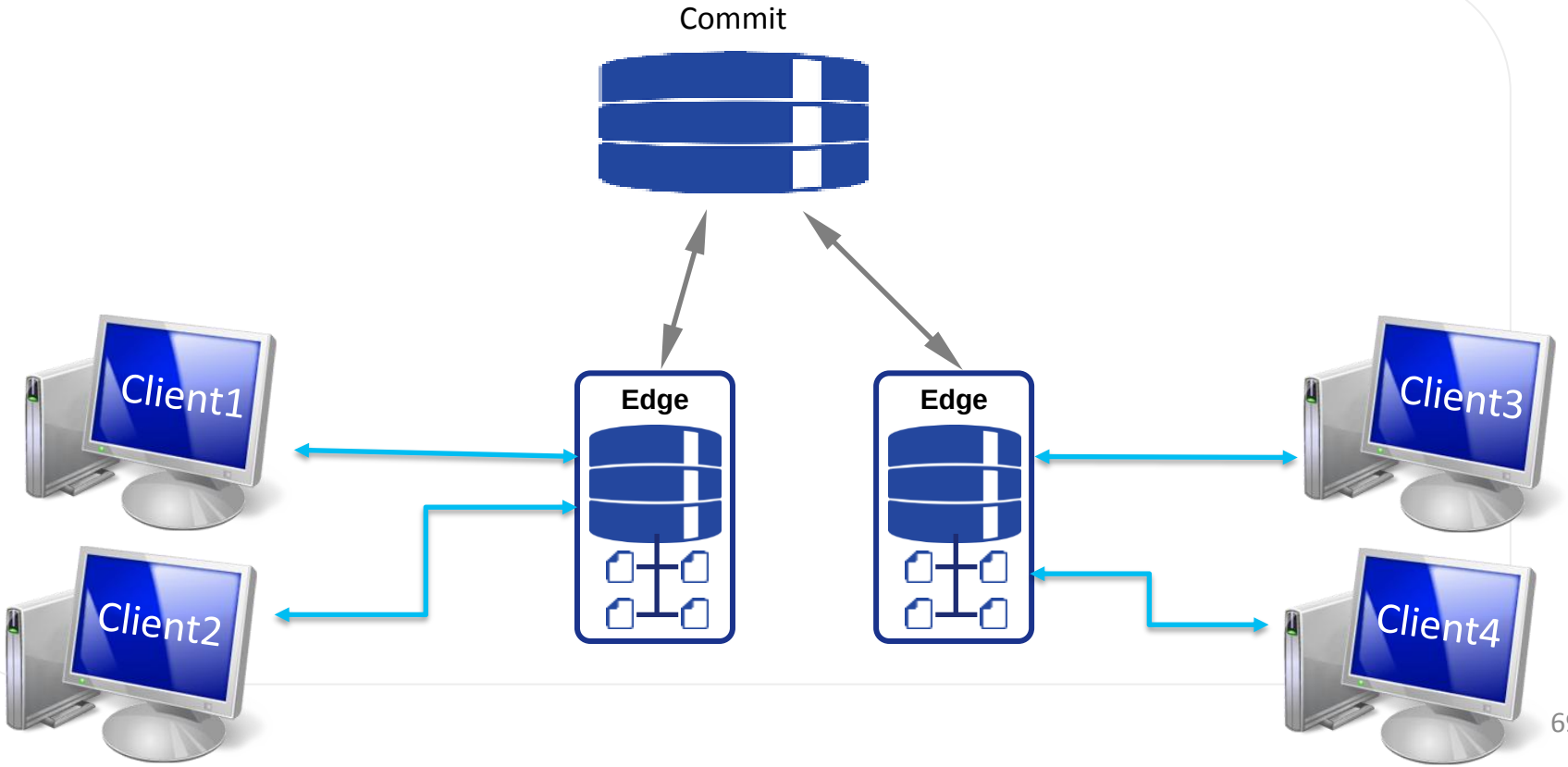
Advanced Enterprise Administration for Perforce

Fully Distributed

Edge/Commit Server Architecture



Edge/Commit Server Architecture



Prepare in the Master

PERFORCE
Version everything.

```
p4 server Edge1

ServerID: Edge1
Name: Edge1
Type: server
Services: edge-server
```

```
p4 configure set Edge1#db.replication=readonly
p4 configure set Edge1#lbr.replication=readonly
```

Equivalent value set via ‘p4 server’ specification:

```
p4 configure set Edge1#rpl.forward.all=1
```

Configuring Edge workspaces

PERFORCE
Version everything.

```
p4 client build-ws-9201
```

```
Client:build-ws-9201
:
ServerID: Edge1
View:
:
```

Edge/Commit Considerations

PERFORCE
Version everything.

- Edge servers contain locally-unique data
  - Generally require backup/recovery
- Information is distributed – you may need to interrogate all edge servers
- Forwarding replicas are simpler
  - Address many needs
  - large db.have is better handled with Edge servers
- Overall user performance is better with Edge servers

- Edge servers used for build farms do not generally require backup

Build-Edge/Commit Considerations

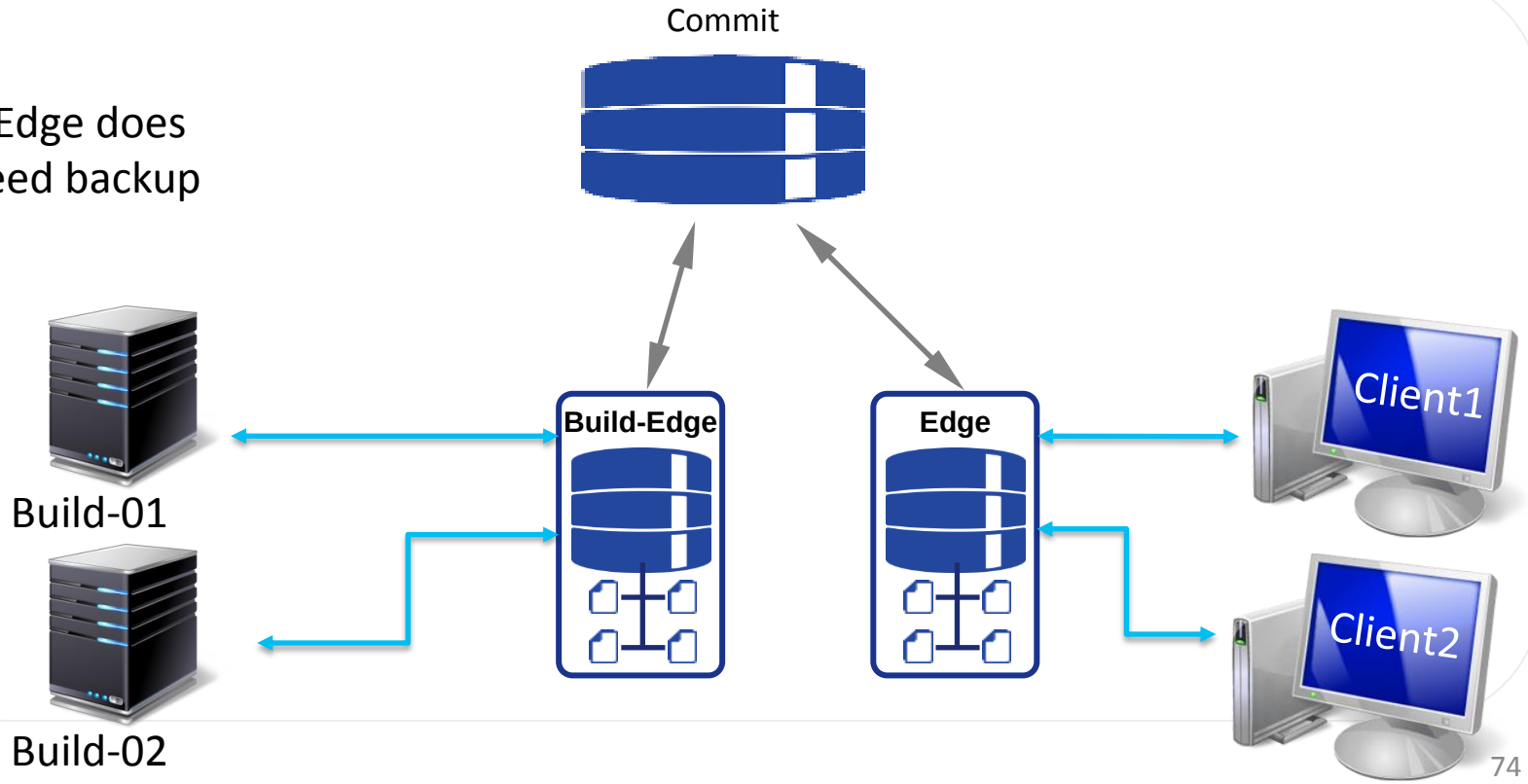
PERFORCE
Version everything.

- Edge servers used for build farms do not generally require backup
- Build data is inherently transient.
- Faster to rebuild from master than to rebuild from scratch
  - Workspaces stored on master
  - ‘Have’ data stored local to Edge
  - Local ‘have’ data not valuable after build is complete

Build-Edge/Commit Server Architecture

PERFORCE
Version everything.

- Build-Edge does not need backup



Exercises

Lab set E5: Forwarding and build replica

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Security

Setting Server Security Level

PERFORCE
Version everything.

- Security settings determine how Perforce Server enforces passwords
- Display security counter value

```
p4 configure show security
security=3 (configure)
```

- Set security counter

```
p4 configure set security=3
```

0	No password required, any password allowed (default)
1	Strong password is required, can be stored in Windows registry
2	Strong password is required, cannot be stored in registry
3	*p4 login* tickets only, no password stored anywhere
4	Level 3 + Edge, replica, proxy & brokers must connect using a service user

Server Security

PERFORCE
Version everything.

- Server security levels (0-4)
  - p4 configure set security=4
- Turn off auto user creation and require authorization to see user list
  - p4 configure set dm.user.noautocreate=2
  - p4 configure set run.users.authorize=1
- Set changelists to restricted by default
  - p4 configure set defaultChangeType=restricted

Connection Protocols

PERFORCE
Version everything.

- TCP
  - Default protocol

```
P4PORT=tcp:p4server:1666
```

- RSH
  - Starts up the server for each request
  - Useful for testing and inetd support

```
P4PORT=rsh:/usr/local/bin/p4d -r $P4ROOT -L $P4LOG -i
```

- SSL
  - SSL encrypted connection when using “ssl:” prefix

```
P4PORT=ssl:p4server:1667
```

RSH connection

PERFORCE
Version everything.

- Starts up a server on client request
- No TCP/IP connection to server
  - Uses stdout/stdin bound to client (with -i option)

- Usage examples:
  - Sidetrack server (specify different log file)
  - Test environments (P4Python, P4Ruby, P4Perl)

SSL Encryption

PERFORCE
Version everything.

- 2012.1+ release support SSL encryption
  - Perforce Server, Perforce Proxy, P4Broker
  - Requires 2012.1+ client (P4, P4V, …)
    - Consider implications with 3rd party integrations
  - If enabled, all clients require SSL connection.
    - Run two P4Ds to offer SSL and non-SSL (one with “ssl:”, one without)
- Client needs fingerprint in its P4TRUST file

```
p4 trust
```

p4 trust

- Client-side command for handling fingerprints
- Uses P4TRUST environment variable (default $Home/.p4trust)

```
p4 trust -h
```

`p4 trust -y`	Accept the fingerprint
`p4 trust -n`	Reject the fingerprint
`p4 trust -f`	Force overwriting of the fingerprint
`p4 trust -l`	List accepted fingerprints
`p4 trust -d`	Delete a fingerprint

SSL Setup

PERFORCE
Version everything.

- P4SSLDIR -> directory with key and certificate

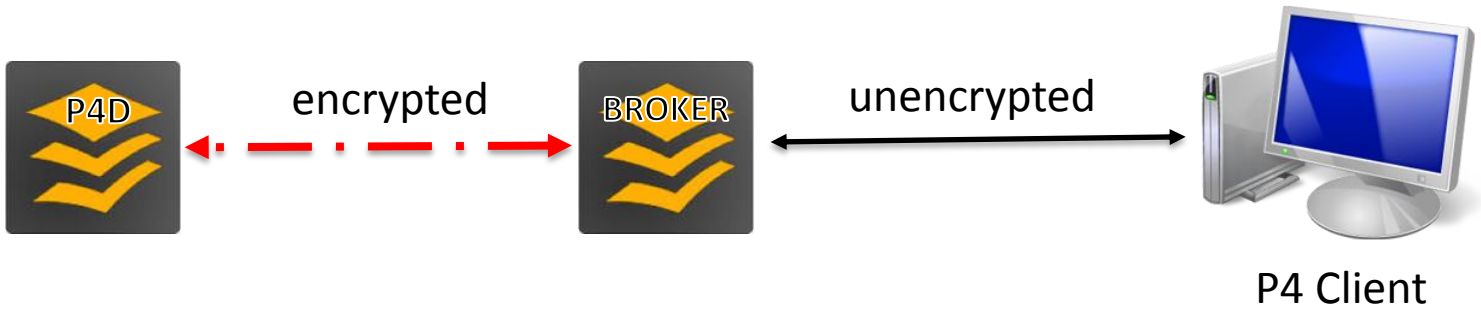
```
cd $P4ROOT
mkdir ssl          # optionally create config.txt
chmod 700 ssl      # drwx------
export P4SSLDIR=ssl
p4d -r . -Gc       # key and certificate
p4d -r . -p ssl:1667
```

- Client needs to accept fingerprint

```
p4 -p ssl:p4server:1667 trust -y
```

Phasing-in SSL encryption with P4Broker

- Use P4Broker
  - P4D runs with SSL encryption enabled
  - P4Broker itself runs unencrypted
  - Allows phasing-in of encrypted connections



Exercises

Lab Set E6: Security

New commands in this chapter:

- `p4d -Gc`
- `p4 trust`

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Advanced Tools

Advanced Tools

PERFORCE
Version everything.

- perfmerge
- perfsplit
- p4-migrate
- Checkpoint surgery
- Conversions – ftp://ftp.perforce.com/perforce/tools

perfmerge

PERFORCE
Version everything.

- **Goal**
  - Merge two Perforce Servers into a single Perforce Server

- **Implementation**
  - Perfmerge tool reads both databases
  - Choice on change merging
    - Append
    - Intersperse and order in time
    - Append with offset

perfsplit

PERFORCE
Version everything.

- **Goal**
  - Extract data from a main server with its exact revision history
  - Split a Perforce Server into two separate Perforce Server

- Implementation
  - Perfsplit reads directly from an existing Perforce Server
  - It uses a splitmap to determine which files are split
    - Same syntax as the label view map
  - Only creates metadata, depot files need to be copied separately

p4migrate

PERFORCE
Version everything.

- **Goal**
  - Migrate a Perforce Server from a case-insensitive to a case-sensitive platform

- **Implementation**
  - Reads a checkpoint to find case inconsistencies
  - Generates a case-fix map
  - Use the map to correct the checkpoint
  - Once the checkpoint is case-consistent it can be used for migration
  - Tool can also be used to rename depot paths
  - Migration from case-sensitive to case-insensitive is not supported

Checkpoint/Journal Format

PERFORCE
Version everything.

- Text file containing journal records
- Each record has a type
  - Checkpoint only has @pv@ entries
- Strings are surrounded by @ symbol
- Each value record refers to
  - A database table
  - The table version

Record	Type
@pv@	Put value = insert
@dv@	Delete value = delete
@rv@	Replace value = update
@vv@	Verify value = select
@ex@	commit
@mx@	flush
@nx@	Journal note

- http://www.perforce.com/perforce/doc.current/schema/

Log Analysis and Reporting

PERFORCE
Version everything.

- Standard Log
  - Log Analyzer
    - Upload your logs
    - Download our tools
  - Track2SQL
- Structured Logs
- Performance monitoring using the log
- Metrics with P4toDB (replication technology)
- Discovering overall trends

Structured Log

PERFORCE
Version everything.

#	Structured Logs	Description
1	`all`	All loggable events (commands, errors, audit, etc...)
2	`commands`	Command events (command start, compute, and end)
3	`errors`	Error events (errors-failed, errors-fatal)
4	`audit`	Audit events (audit, purge)
5	`track`	Command tracking (track-usage, track-rpc, track-db)
6	`user`	User events; one record every time a user runs p4 logappend.
7	`events`	Server events (startup, shutdown, checkpoint, journal rotation, etc.)
8	`integrity`	Replication errors/events

Structured Logs

- Enable specific structured logs with:

```
p4 configure set serverlog.file.n=logtag.csv
p4 configure set serverlog.maxmb.n=1024
p4 configure set serverlog.retain.n=45
```

- Enabling all structured logging files can consume considerable space and impact performance.
- Structured logs are automatically rotated on checkpoint, journal rotation, exceeding size limit, or when ‘p4 logrotate’ is run.

Conclusion

PERFORCE
Version everything.

- Database schema is public
- There are tools that can use the checkpoint or the database directly
- Handle with care

- Ask Perforce support or consulting if you are not sure

Exercises

Lab Set E7: Structured Logs

New commands in this chapter (samples):

- `p4 configure set serverlog.file.n=errors.csv`
- `p4 configure set serverlog.maxmb.n=30Mb`
- `p4 configure set serverlog.retain.n=45`
- `p4 logappend`
- `p4 logrotate`

PERFORCE
Version everything.

Advanced Enterprise Administration for Perforce

Scripting

Preliminary Decisions

PERFORCE
Version everything.

- Uses of scripts
- Choosing the interface
- Setting Environment Variables
- User Authentication

Uses of scripts

PERFORCE
Version everything.

- Reporting tools
- Daemons and recurring processes
- Wrappers for Perforce commands
- Triggers
- Workflow and policy enforcement
- P4V customization (P4JsAPI)
- P4Broker
- Legacy SCM data import

Typical workings of a script

PERFORCE
Version everything.

- Data processing in batches
  - Retrieve information such as files or changes
  - Process the data in the script
  - Potentially update Perforce
- Form handling
  - Retrieve a form such as a client workspace
  - Modify the form in the script
  - Update the form in Perforce

Workflow and Policy enforcement

PERFORCE
Version everything.

- Triggers
  - Submit/Shelving triggers
  - Authentication triggers
  - Form triggers
  - Archive triggers
  - Fix triggers

- P4Broker
  - Block, redirect or modify commands

Choosing the interface

PERFORCE
Version everything.

- Wrap P4 command
  - \+ Simple solution that will run everywhere
  - \+ Batch scripting built into the OS and requires no installation
  - – Requires parsing of output
- APIs
  - \+ Language-specific integration
  - \+ Extendable
  - \+ Performance (reduced connection overhead)
  - – Requires installation (and/or build/compilation)

API's Available for Scripting

PERFORCE
Version everything.

- Programming Languages
  - C++
  - P4Java
  - Perforce Objective-C
  - Perforce .NET
- Derived APIs (C++ API wrappers)
  - P4Python
  - P4Perl
  - P4Ruby
  - P4PHP

http://www.perforce.com/product/components/apis

Wrapping the command line client P4

PERFORCE
Version everything.

- Command line returns lines of text

```
p4 describe -s 13
Change 13 by sknop@alita on 2010/03/02 12:58:51

 Branching foo from bar.
 Test branch only.

Affected files ...

... //depot/tests/foo#1 branch
```

Capture errors, warnings and messages

PERFORCE
Version everything.

- Use -s to precede each output line with “info” or “error”

```
p4 -s sync ...
info: //depot/foo#3 - updating /client/foo
error: Can't clobber writable file /client/foo
exit: 1
```

Tagging output: Command line and API

PERFORCE
Version everything.

- Format output by using -ztag

```
p4 -ztag clients
... client bruno_ws
... Update 1104271684
... Access 1104340062
... etc.
```

- Perforce API based on tagged data output

Form handling: bypassing an editor

PERFORCE
Version everything.

- Redirect to standard output

```
p4 change -o
```

- Read from standard input

```
p4 submit -i
```

- Submit without invoking an editor

```
p4 submit -d "Fixed off-by-one error."
```

- Example: Create a client workspace without invoking an editor

```
p4 client -o | p4 client -i
```

Setting the environment for scripts

PERFORCE
Version everything.

- Command line flags

```
p4 -p server:1666 -u script_user -c script_ws info
```

- P4CONFIG (next slide)
- Environment and registry variables

- Recommendation:
  - Use P4CONFIG
  - Set P4CONFIG in the scripts to make sure it is set in the environment
  - This will keep scripts independent of Perforce Server and location

P4CONFIG

PERFORCE
Version everything.

- P4CONFIG points to a file name

```
p4 set P4CONFIG=P4Config.txt
export P4CONFIG=/p4/scripts/.p4config
```

- File usually located in the workspace root or scripts folder

- File contains the Perforce variables

```
P4PORT=server:1666
P4CLIENT=script_ws
P4USER=script_user
```

User authentication for scripts

PERFORCE
Version everything.

- `p4 login`
  - Works for all Perforce Server security levels
  - Works if Perforce is integrated with AD
  - Works if Perforce is integrated with SSO

- Either: Store password in local (hidden/restricted access) file

```
p4 login < /p4/scripts/.password
```

- Or: Use ever-lasting ticket (ideally with separate P4TICKETS file)

Use a group to extend session

PERFORCE
Version everything.

```
p4 group scripts
```

```
Group:    scripts
MaxResults:     1000000
Maxscanrows:    5000000
MaxLockTime:      30000
Timeout:      unlimited
Subgroups:
Owners:
                bruno
Users:
                script_user
```

P4TICKETS

- P4TICKETS points to a ticket file

```
export P4TICKETS=/p4/scripts/.script_p4tickets
```

- Important when scripts may be run as a different user (default value is home directory which is different per user)
- This will provide safety from someone accidently logging out a script user
  - Beware of p4 –u script_user logout –a
    - Invalidates all tickets for this user

PERFORCE
Version everything.

Questions?

The End

PERFORCE
Version everything.

All Perforce manuals and technical notes are available at
www.perforce.com

Follow and participate with the Perforce Community and Forums at
www.perforce.com/community
workshop.perforce.com

Report problems and get technical help from support@perforce.com