Introduction to Helix for Users

PERFORCE

Introduction

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

Course Contents (Day 1)

- Overview
- Help!
- File Operations
- File Reporting and Revision Specifiers
- Changelist Management
- Handling File Conflicts

Course Contents (Day 2)

- Client Workspace Management
- Branching and Merging
- Streams
- Labels
- Defect Tracking

Notation used herein

- **p4 command and flags or *variables:***

  p4 -p *port* command

- Items of note in output
- Examples of **commands** in text
- Sample output:

```
$ p4 ping -c 1000 -s 5120000
2.24s for 1000 messages of 5120000 characters
```

Introduction to Helix for Users

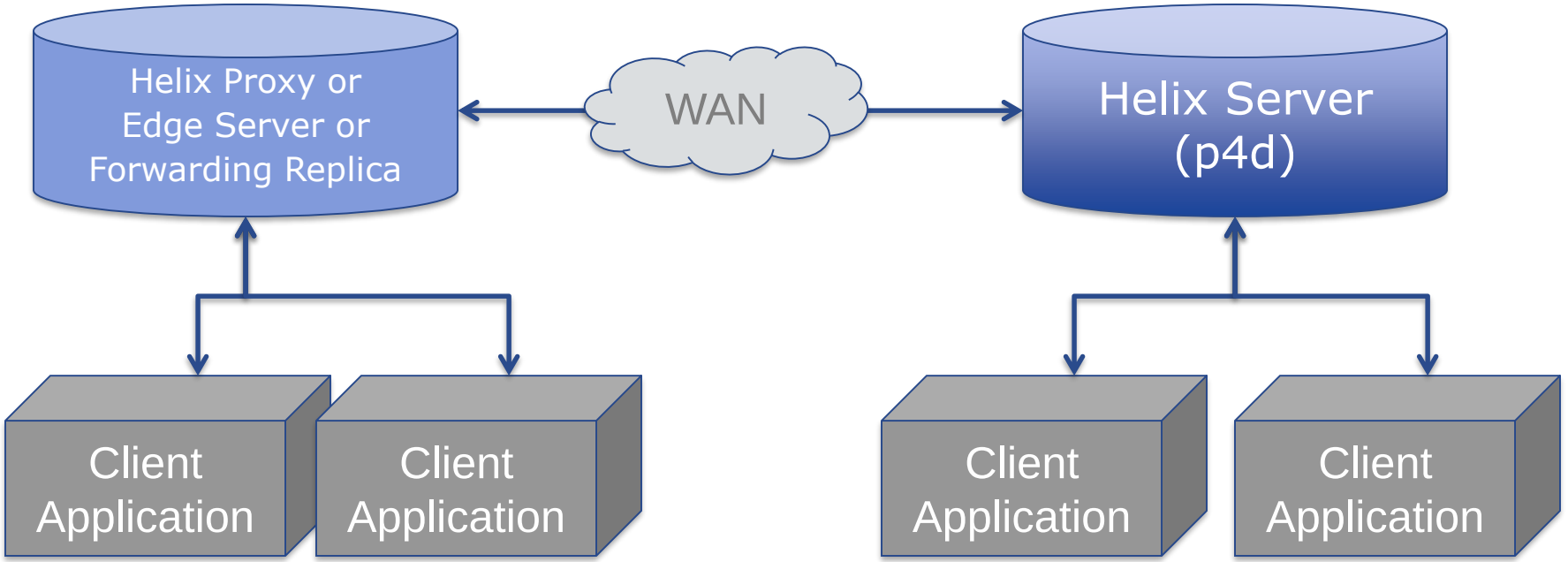
Overview

PERFORCE

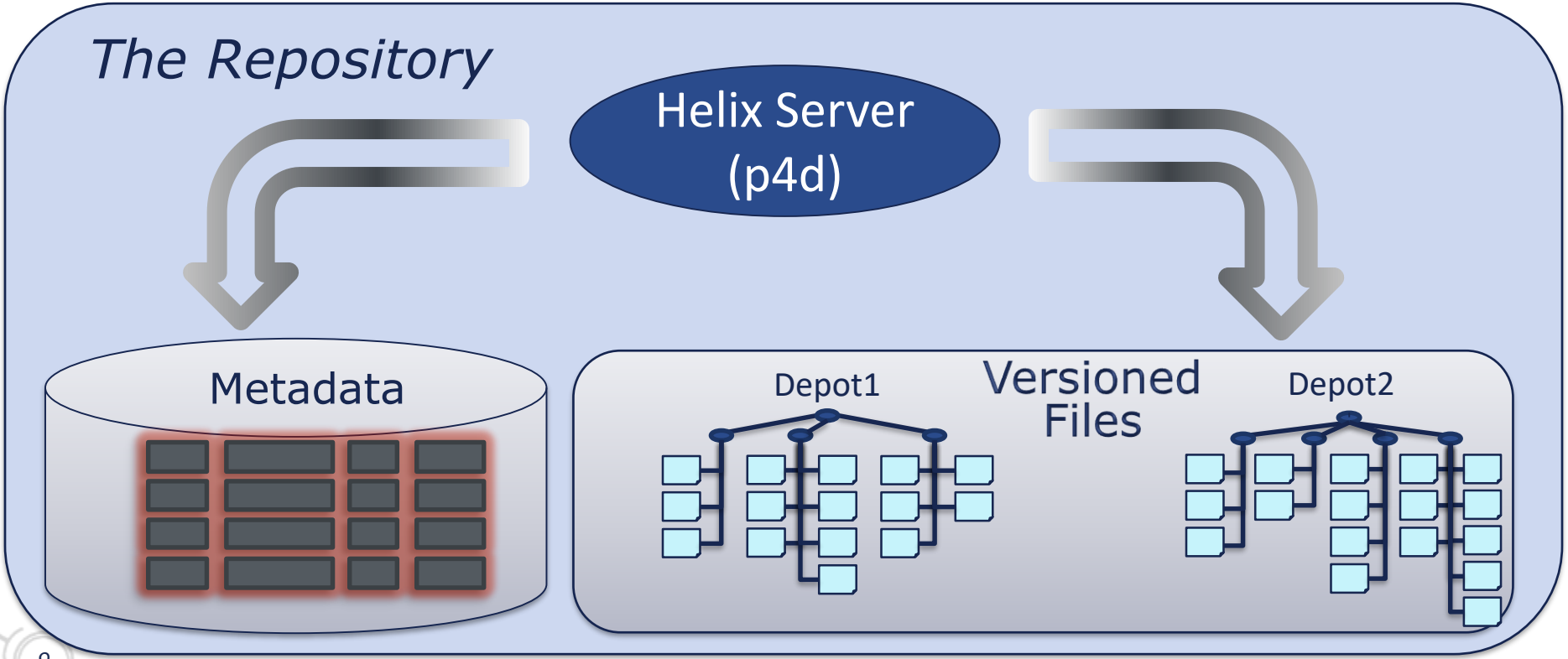
Overview

- Client/Server Model
- Server
- Client Programs
- Changelists
- Client...

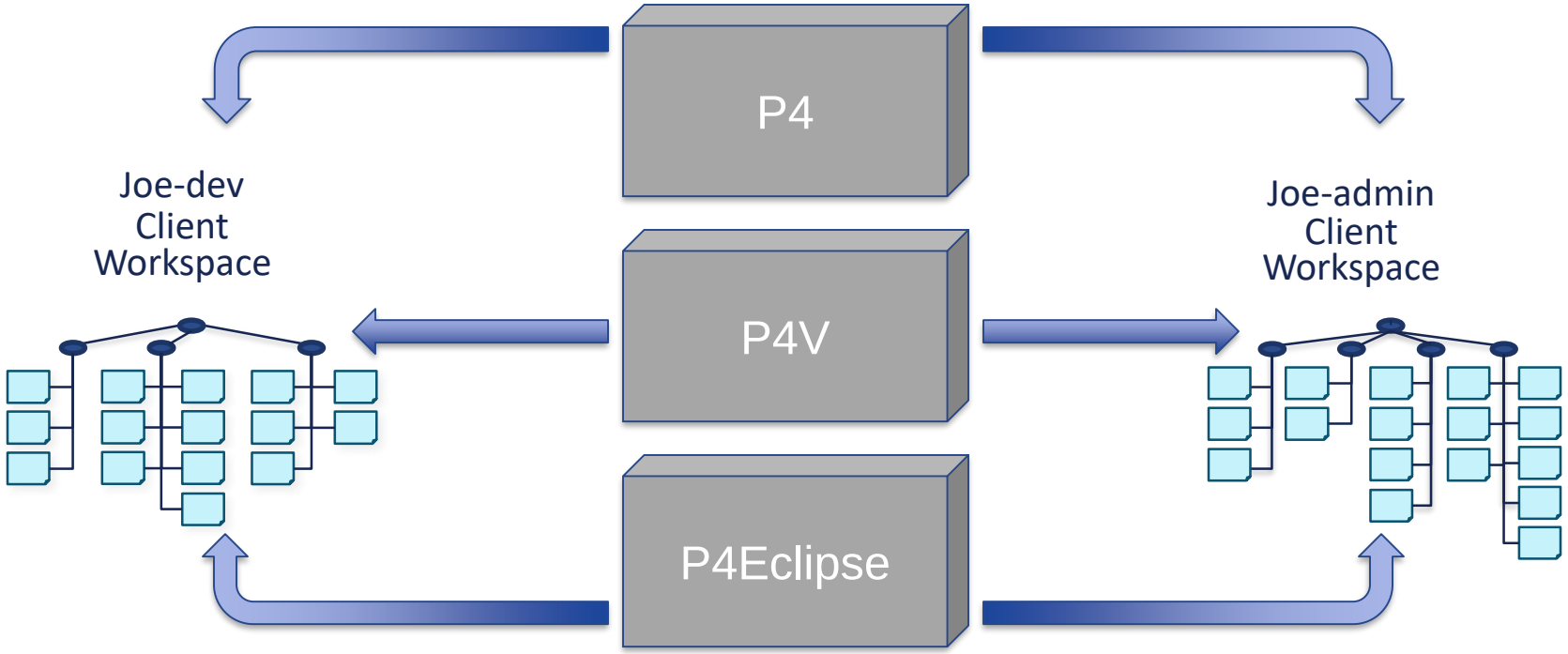
Helix Client-Server model



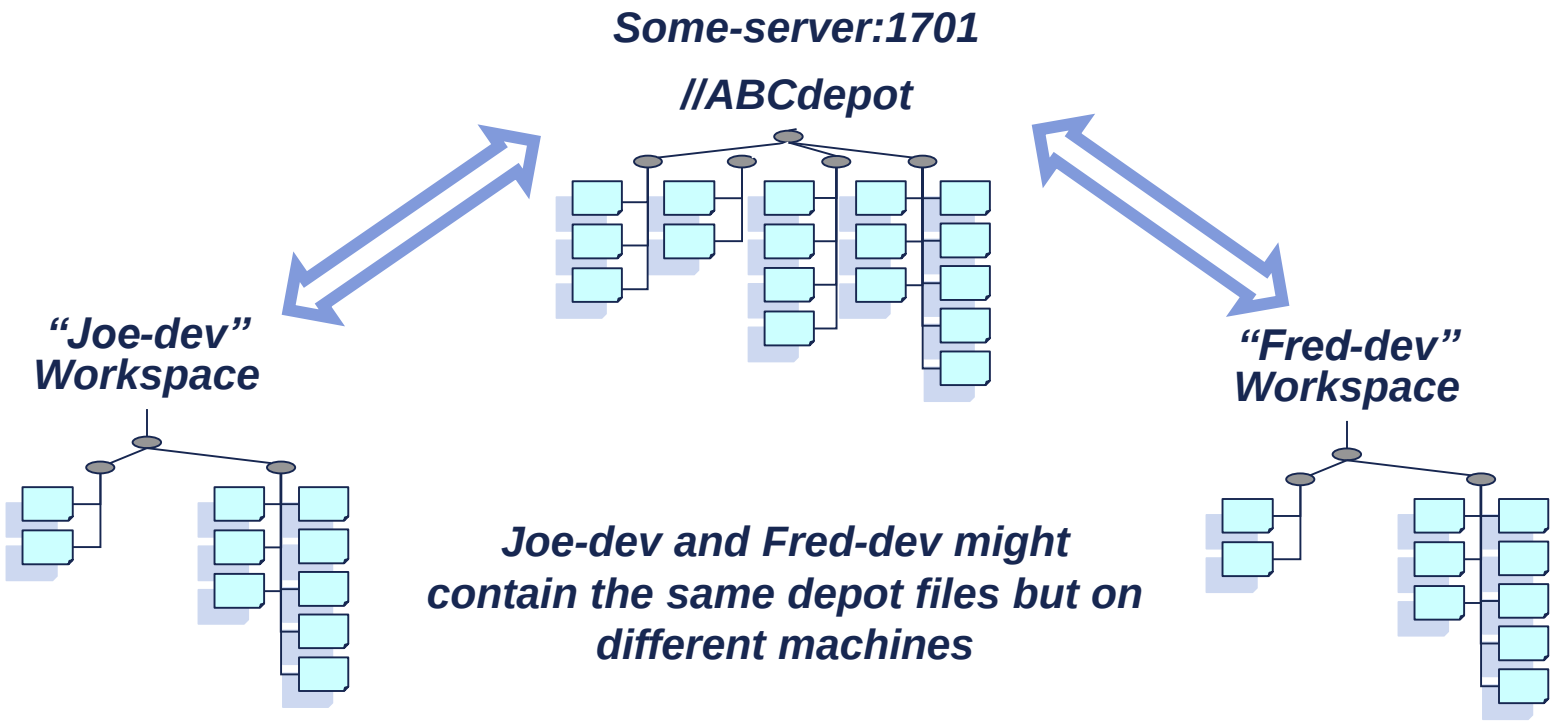
Helix Server



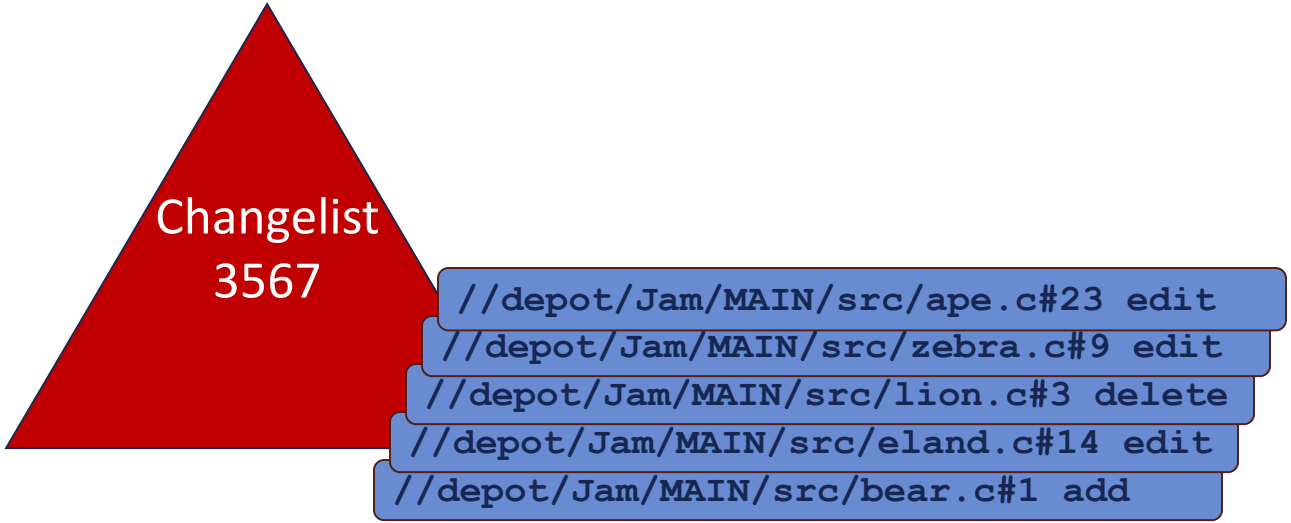
Helix Client applications



Server & Workspaces



Changelists



Client ...

- Client machine
- Client programs
- Client workspaces

Showing Connection Information

```
$ p4 info
User name: bruno
Client name: bruno_ws
Client host: bruno-win.perforce.com
Client root: c:\work
Current directory: c:\work\Jam\MAIN\src
Client address: 103.14.25.211:1719
Server address: london:1666
Server root: /usr/perforce
Server date: 2015/03/26 11:21:36 +0100 BST
Server uptime: 382:14:58
Server version: P4D/DARWIN90U/2011.1/... (2012/01/25)
Server license: Perforce 200 users...
Case Handling: insensitive
```

Commands in this Chapter

- **`p4 info`**

Introduction to Helix for Users

Help!

PERFORCE

Help!

- Command Summary Help
- Help for a Specific Command
- Online Perforce Documentation
- E-mail Perforce
- Perforce Forums

Command Summary Help

`p4 help`

`p4` is Perforce's client tool for the command line.

Try:

`p4 help`	list most common commands
`p4 help commands`	list all standard commands
`p4 help command`	help on a specific command
`...etc...`	

Perforce Commands

p4 help commands

```
Perforce client commands:

   add      Open a new file to add it to the depot
   admin    Perform administrative operations on the server
   annotate Print file lines along with their revisions
   archive  Archive obsolete revisions to archive depots
   branch   Create or edit a branch specification
   branches Display list of branches
   ...etc...
```

Help for a Specific Command

```
p4 help delete
```

```
delete -- Open an existing file to delete it from the depot

p4 delete [ -c changelist# ] [ -n -v ] file ...

    Opens a depot file for deletion.

    If the file is synced in the client workspace, it is
    removed.  If a     pending changelist number is specified
    using with the -c flag, the file is opened for delete in
    that changelist. Otherwise, it is opened in the default
    pending changelist.
```

Perforce Documentation

- Introducing Perforce
- P4 User's Guide
- Perforce Command Reference
- Perforce System Administrator's Guide
- Release Notes
- Knowledge Base
- White Papers

Help via E-mail

- E-mail Perforce at:
  - support@perforce.com
  - sales@perforce.com
  - consulting@perforce.com
- Perforce Forums
  - Subscribe at http://forums.perforce.com/
  - Exchange ideas with other Perforce users

Commands in this Chapter

- **`p4 help`**

Introduction to Helix for Users

Basic File Operations

PERFORCE

Basic Operations

- Populating a Client Workspace
- Editing, Adding and Deleting Files
- Moving Files
- Reverting Files
- Assessing Current Status
- Submitting Changes

Populating a Client Workspace

```
p4 sync
//depot/Jam/MAIN/src/README#26 - added as c:\work\Jam\MAIN\src\README
//depot/Jam/MAIN/src/regexp.c#2 - added as c:\work\Jam\MAIN\src\regexp.c
//depot/Jam/MAIN/src/regexp.h#1 - added as c:\work\Jam\MAIN\src\regexp.h
//depot/Jam/MAIN/src/RELNOTES#77 - added as c:\work\Jam\MAIN\src\RELNOTES
//depot/Jam/MAIN/src/rules.c#5 - added as c:\work\Jam\MAIN\src\rules.c
...etc...
```

Adding New Files

```
cd \work\Jam\MAIN\src
notepad alias.c
notepad check.c

p4 add alias.c check.c
//depot/Jam/MAIN/src/alias.c#1 - opened for add
//depot/Jam/MAIN/src/check.c#1 - opened for add
```

Editing Existing Files

```
cd \work\Jam\MAIN\src

p4 edit rules.c rules.h
//depot/Jam/MAIN/src/rules.c#5 - opened for edit
//depot/Jam/MAIN/src/rules.h#2 - opened for edit

notepad rules.c

notepad rules.h
```

Deleting Files

```
p4 delete exec.c //depot/Jam/MAIN/dir.c
//depot/Jam/MAIN/src/exec.c#2 - opened for delete
//depot/Jam/MAIN/dir.c#3 - opened for delete
```

Moving Files

```
p4 edit filemac.c
//depot/Jam/MAIN/src/filemac.c#3 - opened for edit

p4 move filemac.c fileosx.c
//depot/Jam/MAIN/src/fileosx.c#1 - moved from
                                  //depot/Jam/MAIN/src/filemac.c#3
```

Undoing Your Changes

```
p4 revert alias.c fileosx.c rules.h //depot/Jam/MAIN/dir.c
     //depot/Jam/MAIN/images/logo.pdf
```

```
//depot/Jam/MAIN/dir.c#3 – was delete, reverted
//depot/Jam/MAIN/images/logo.pdf#none – was add, abandoned
//depot/Jam/MAIN/src/alias.c#none - was add, abandoned
//depot/Jam/MAIN/src/rules.h#2 - was edit, reverted
//depot/Jam/MAIN/src/filemac.c#3 - was move/delete, reverted
//depot/Jam/MAIN/src/fileosx.c#none - was move/add, deleted
```

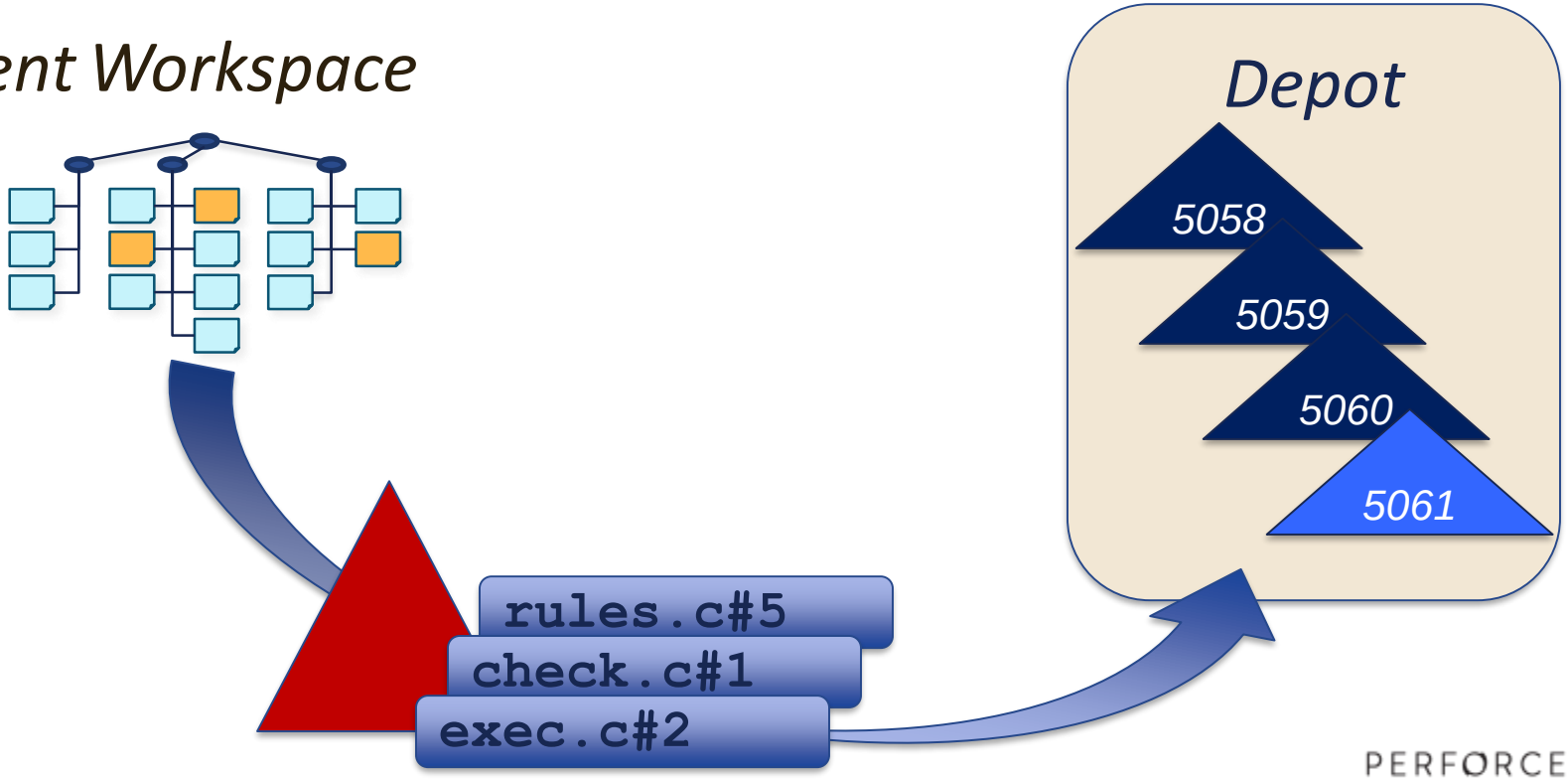
Which Files Were Opened?

```
p4 opened
//depot/Jam/MAIN/src/check.c#1 - add default change (text)
//depot/Jam/MAIN/src/exec.c#2 -  delete default change (text)

p4 opened -u bob
//depot/Jam/MAIN/src/make.c#5 - edit default change (text) by bob@bo
//depot/Jam/MAIN/src/rcp.c#7 - edit default change (text)  by bob@bo
//depot/Jam/MAIN/src/trans.h#9 - edit default change (text) by bob@j
```

What Changes Have You Made?

```
p4 diff rules.c


==== //depot/Jam/MAIN/src/rules.c#5 -
  c:\work\Jam\MAIN\src\rules.c ===
76c76
<         TARGET target, t = &target;
---
>         TARGET target, *t = &target;
```

Submitting a changelist



Submitting a Changelist

p4 submit

```
Change:   new
Description:
     Added new check function
Files:
 //depot/Jam/MAIN/src/check.c   # add
 //depot/Jam/MAIN/src/exec.c    # delete
 //depot/Jam/MAIN/src/filemac.c # move/delete
 //depot/Jam/MAIN/src/fileosx.c # move/add
 //depot/Jam/MAIN/src/parse.c   # edit
 //depot/Jam/MAIN/src/rules.c   # edit
```

Previewing Workspace Update

```
p4 sync -n
//depot/Jam/MAIN/src/README#27 - updating c:\work\Jam\MAIN\src\README
//depot/Jam/MAIN/src/regexp.c#3 - updating c:\work\Jam\MAIN\src\regexp.c
//depot/Jam/MAIN/src/regexp.h#2 - updating c:\work\Jam\MAIN\src\regexp.h
//depot/Jam/MAIN/src/RELNOTES#79 - updating c:\work\Jam\MAIN\src\RELNOTES
//depot/Jam/MAIN/src/varexp.c#8 - added as c:\work\Jam\MAIN\src\varexp.c

...etc...
```

New Commands in this Chapter

- `p4 sync`
- `p4 edit`
- `p4 add`
- `p4 delete`
- `p4 move`
- `p4 revert`
- `p4 opened`
- `p4 diff`
- `p4 submit`

Introduction to Helix for Users

More on File Operations

PERFORCE

More on File Operations

- Supported Wildcards
- Referencing files
- File types
- Reconcile Offline Work
- “Proactive Checkout” vs. “Edit First” Workflows

Supported Wildcards

- Two types

```
p4 sync ...
p4 sync //depot/Jam/MAIN/...
p4 sync //depot/Jam/R1.0/....txt
p4 sync //depot/Jam/MAIN/doc/J*
p4 sync //depot/Jam/MAIN/src/*
```

- Local vs. depot expansion

```
p4 sync *.c
p4 sync "*.c"
```

Referencing Files

- Depot Syntax

```
p4 sync //depot/Jam/MAIN/src/util.c
```

- Local Syntax

```
p4 sync util.c
p4 sync c:\work\Jam\MAIN\src\util.c
p4 sync ..\src\util.c
```

File Types

- Base file types

```
text                    symlink
binary                  unicode                 utf16
```

- Workspace modifiers

```
+x  - executable                +w  - always writable
+l  - exclusive open            +k  - RCS keyword
+m  - sync vs. submit modtime
```

- Server storage attributes

```
+Sn  -  last 'n' revisions      +D -  deltas
+C   -  compressed              +F -  full file
```

Setting File Type

- Use the `-t` flag:

```
p4 add -t binary *.pdf
p4 edit -t text genfiles
```

- For already opened files, use `p4 reopen`

```
p4 reopen -t binary+lm doit
p4 reopen -t +w ....txt
```

Reconcile Offline Work

- See what files were changed offline in a directory:

```
cd c:\sean-jam\Jam\MAIN\src
```

```
p4 status -m
```

- Open locally changed files for add, edit, delete, and move:

```
p4 rec
```

Reconcile Offline Work (Alternate)

```
cd c:\sean-jam\Jam\MAIN\src

dir /s/b/a-d          | p4 -x - add

p4 diff -sd ...       | p4 -x - delete

p4 diff -se ...       | p4 -x - edit -t auto
```

“Proactive Checkout” Workflow

- Use `noallwrite` workspace option.
- Optionally use **`p4 change`** to create changelist first.
- Use **`p4 add`**, **`p4 edit`**, **`p4 delete`**, and **`p4 move`** before making changes to local files.
- Best option when working with very large workspaces.
- Generally considered more formal, disciplined workflow.
- Default unless using DVCS features.
- Similar to ClearCase.

“Edit First” Workflow

- Use `allwrite` workspace option.
- Make changes to local files first.
- Use **`p4 status`** and **`p4 rec`** (per directory or from the top of the tree) to prepare files to commit.
- Generally considered less formal workflow.
- Beware: Easy to forget files in other directories if not working at the top of the tree.
- Default when using Perforce DVCS features.
- Similar to Subversion, Git.

Summary: “Proactive Checkout” vs. “Edit First” is largely a matter of personal preference.

New Commands in this Chapter

- `p4 add`
- `p4 edit -t`
- `p4 reopen -t`
- `p4 reconcile`
- `p4 status`

Introduction to Perforce for Users

File Reporting

PERFORCE

File Reporting

- Depot File Information
- Directory information
- Changelist Information
- What’s in your workspace
- Search for patterns in files
- Show file contents with revisions

Depot File Information

- List files in a depot

```
p4 files //depot/Jam/MAIN/...
//depot/Jam/MAIN/utils.c#3 - edit change 5121 (text)
...etc...
```

- List revision history of a file

```
p4 filelog utils.c
//depot/Jam/MAIN/utils.c
... #3 change 5121 edit ... 'add reuse feature'
... #2 change 5119 edit ... 'fix startup bug'
... #1 change 4967  add  ... 'add Gizmo src files'
```

Directory Information

- List all directories under a given path

```
p4 dirs //depot/Jam/*
//depot/Jam/MAIN
//depot/Jam/REL2.1
//depot/Jam/REL2.2
```

- Artifact of file path
- Directories are not versioned

Changelist Information

- List all numbered changelists

```
p4 changes -m 9
Change 5120 on 2014/06/15 by bruno@ws ...
Change 5119 on 2014/06/15 by bruno@dyn_ws ...
...etc...
```

- Restrict to a path or file

```
p4 changes //pb/...
```

- Restrict to user

```
p4 changes -u bruno
```

What's in Your Workspace?

```
p4 have
//depot/Jam/MAIN/src/Build.com#7 - c:\work\Jam\MAIN\src\Build.com
//depot/Jam/MAIN/src/Build.mpw#1 - c:\work\Jam\MAIN\src\Build.mpw
//depot/Jam/MAIN/src/command.c#8 - c:\work\Jam\MAIN\src\command.c
...etc...
```

Search for Patterns in Files

```
p4 grep -e "rules.h" //depot/Jam/...
//depot/Jam/MAIN/src/command.c#8:
  # include "rules.h"
//depot/Jam/MAIN/src/compile.c#25:
  # include "rules.h"
//depot/Jam/MAIN/src/compile.c#25:
  * directly to the rule (as defined in rules.h).
//depot/Jam/MAIN/src/headers.c#5:
  # include "rules.h”
...etc...
```

Show file contents with revisions

```
p4 annotate jam.c

//depot/Jam/MAIN/src/jam.c#35 - edit change 627 (text)
1:        /*
1:    *
16:        *      Copyright 1993, 1996 Chris.
...etc...
```

```
p4 annotate -a -c jam.c

//depot/Jam/MAIN/src/jam.c#35 - edit change 627 (text)
1-627:    /*
1-627:     *
1-1:            *         Copyright 1993 Chris.
30-248:    *         Copyright 1993, 1995 Chris.
...etc...
```

- P4V's Time-Lapse View gives an excellent graphical view

New Commands in this Chapter

- `p4 files`
- `p4 filelog`
- `p4 changes`
- `p4 have`
- `p4 grep -e`
- `p4 annotate`

Introduction to Helix for Users

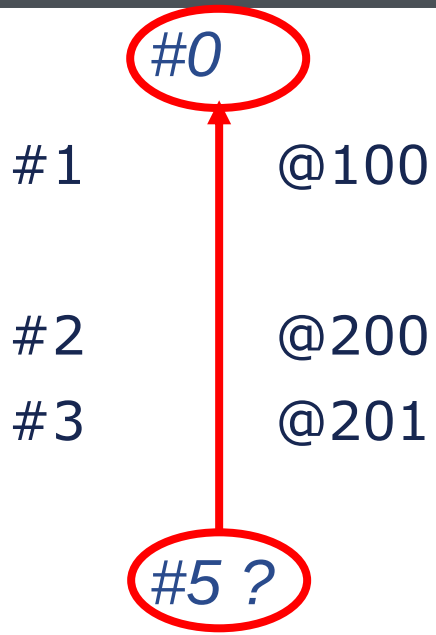
Revision Specifiers

PERFORCE

Revision Specifiers

- Revision specifiers
- Revision ranges
- Comparing different revisions
- Backing out a changelist

Syncing a file – what happens?



@50 ?

@150 ?

@9999999 (doesn't exist) ?

What are Revision Specifiers?

- Identify a file or group of files

Changelist	@5123	@5179	@5234	@5259	@5310
file1	#1	#2		#3	#4
file2		#1	#2		#3
file3	#1			#2	

- A changelist specifier can refer to:
  - The set of files contained in the changelist
  - A point in time in the life of your repository (like a label)

Revision Specifier syntax

Type	Syntax	Used by
Revision number	#7	CLI / P4V
Changelist number	@30	CLI
Workspace name	@phil_ws	CLI
Label name	@jerrys_label	CLI
Latest revision	#head	CLI / P4V
Have revision	#have	CLI / P4V
Non-existent revision	#none or @0	CLI / P4V

Why Use Revision Specifiers?

- Restrict the scope of a command:
  - Sync to a particular revision

```
p4 sync //depot/Jam/MAIN/rules.h#12
```

  - List files updated up to a known changelist

```
p4 files //depot/Jam/...@5234
```

  - List changes submitted up to a known date

```
p4 changes //depot/Jam/MAIN/...@2015/04/21
```

Revision Specifiers - CLI

```
p4 sync utils.c#2
p4 sync utils.c@431
p4 sync @431
p4 sync ...@431
p4 changes //depot/Jamgraph/DEV/gizmo/*.c@431
p4 sync @janslabel
p4 files //depot/Acme-api/...#have
p4 sync //depot/notes/....txt
p4 sync #none
p4 sync util.c@2014/05/29
p4 files //depot/Jam/...@2014/05/29:15:37:00
```

Revision ranges – CLI only

- Limiting commands to a revision range:

```
p4 changes abc.c@20,@32
p4 changes xyz.c@rel1,@rel2
p4 files //depot/api/...@431,@431
p4 files //depot/api/...@=431
```

- Some commands accept revision ranges:

```
p4 changes    p4 merge    p4 sync
p4 jobs       p4 files    p4 print
p4 fixes
```

Comparing different revisions

```
p4 diff2 hash.c#1 hash.c#2

==== //depot/Jam/MAIN/src/hash.c#1 -
     //depot/Jam/MAIN/src/hash.c#2 ====
2c2,4
<  * Copyright 1993 Christopher Seiwald.
---
>  * Copyright 1993, 1995 Christopher Seiwald.
>  *
>  * This file is part of Jam - see jam.c for
     Copyright information.
```

Which Files Have Changed?

```
p4 diff2 -q //depot/...@231 //depot/...@250

==== //depot/Jam/MAIN/src/jam.c#13 (text) -
      //depot/Jam/MAIN/src/jam.c#15 (text) ==== content
 ==== //depot/Jam/MAIN/src/jam.h#17 (text) -
      //depot/Jam/MAIN/src/jam.h#18 (text) ==== content
 ==== <none> - //depot/Jam/MAIN/src/Jambase.ps#1 ====
 ==== <none> - //depot/Jam/MAIN/src/Jamfile.ps#1 ====
```

Recovering a Deleted File

- Sync the revision to be re-added:

```
p4 sync parse.h@5065
```

Or

```
p4 sync parse.h#9
```

- Open the file for add

```
p4 add parse.h
```

- Submit

New Commands in this Chapter

- **`p4 diff2`**

Introduction to Helix for Users

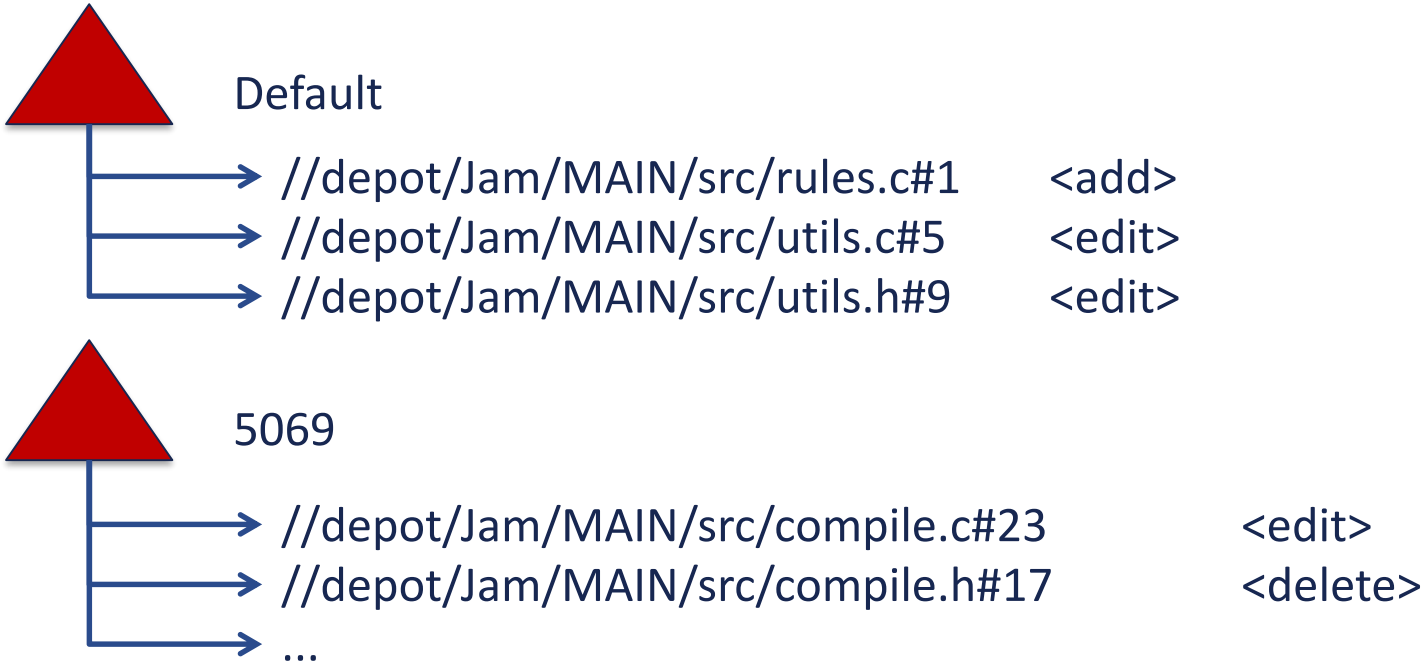
Changelist Management

PERFORCE

Changelist Management

- Managing pending changelists
- Shelving opened files
- Showing changelist detail

Multiple Pending Changelists



Creating a New Changelist

p4 change

```
Change: new
Client: bob-jam
User:   bob
Status: pending
Description:
    Fix off-by-one error.
Files:
    //depot/Jam/MAIN/src/compile.c #edit
```

```
Change 5069 created with 1 open file(s).
```

Editing a Pending Changelist

```
p4 change 5069
```

```
Change: 5069
Client: bob-jam
User:   bob
Status: pending
Description:
    Fix off-by-one error. Use UTC time.
Files:
    //depot/Jam/MAIN/src/compile.c #edit
```

Using Numbered Changelists

- Open files into specific changelist

```
p4 edit -c 5061 //cl/Jam/MAIN/src/make.c
p4 add -c 5062 hash.h hash.c
p4 delete -c 5062 //depot/Jam/MAIN/src/filent.c
```

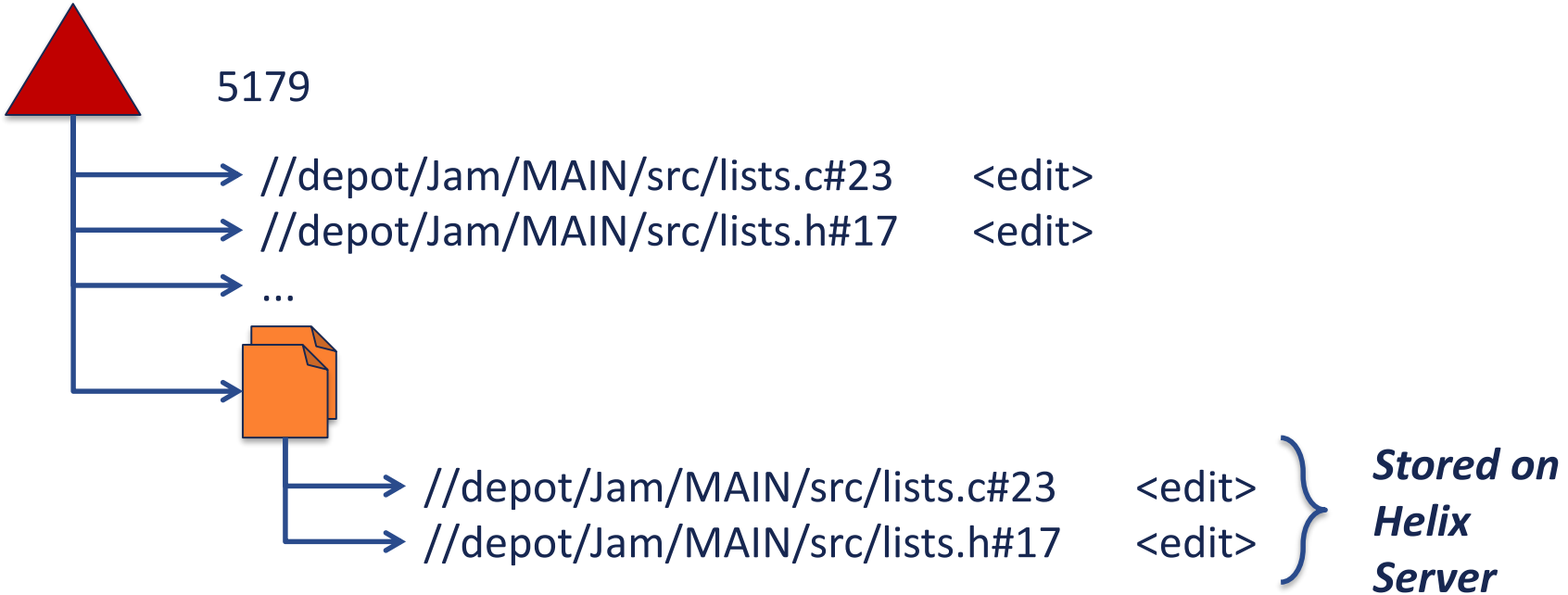
- Move open files between changelists

```
p4 reopen -c 5061 *.sh
p4 reopen -c default //...
```

Submitting a Numbered Changelist

```
p4 submit -c 5069
Submitting change 5069.
Locking 3 files …
add //depot/Jam/MAIN/src/newstr.c#1
delete //depot/Jam/MAIN/src/execmac.c#3
edit //depot/Jam/MAIN/src/rules.c#7
Change 5069 renamed 5072 and submitted
```

Shelved Changes



Shelving Opened Files

```
p4 shelve
```

```
Change: new
Client: bob-jam
User:   bob
Status: new
Description:
    Shelving current edits
Files:
   //depot/Jam/MAIN/src/lists.c # edit
   //depot/Jam/MAIN/src/lists.h # edit
```

Accessing Shelved Files

```
p4 unshelve -s 5179
//depot/Jam/MAIN/src/lists.c#6 - unshelved,
    opened for edit
//depot/Jam/MAIN/src/lists.h#4 - unshelved,
    opened for edit
```

```
p4 diff lists.c@=5179
==== //depot/Jam/MAIN/src/lists.c#45 -
     c:\earl-os2\Jam\MAIN\src\lists.c ====
37,39d36
< # ifdef NTFS
< # include "daylight.h"
< # endif
```

Deleting Shelved Files

```
p4 changes -s shelved
Change 5179 on 2011/11/06 by bob@bob-jam *pending*
  'Shelving current edits. '

p4 shelve -d -c 5179
Shelve 5179 deleted.
```

Showing Changelist Detail

```
p4 describe -s 5185

Change 9185 by esau@esau-dev on 2014/07/02 10:15:38

        Windows 8 changes.

Affected files ...

... //depot/Jam/MAIN/src/Jambase#52 edit
... //depot/Jam/MAIN/src/Jamfile#18 edit
... //depot/Jam/MAIN/src/RELNOTES#15 edit
```

New Commands in this Chapter

- `p4 change`
- `p4 shelve`
- `p4 unshelve`
- `p4 describe`

Introduction to Helix for Users

Handling File Conflicts

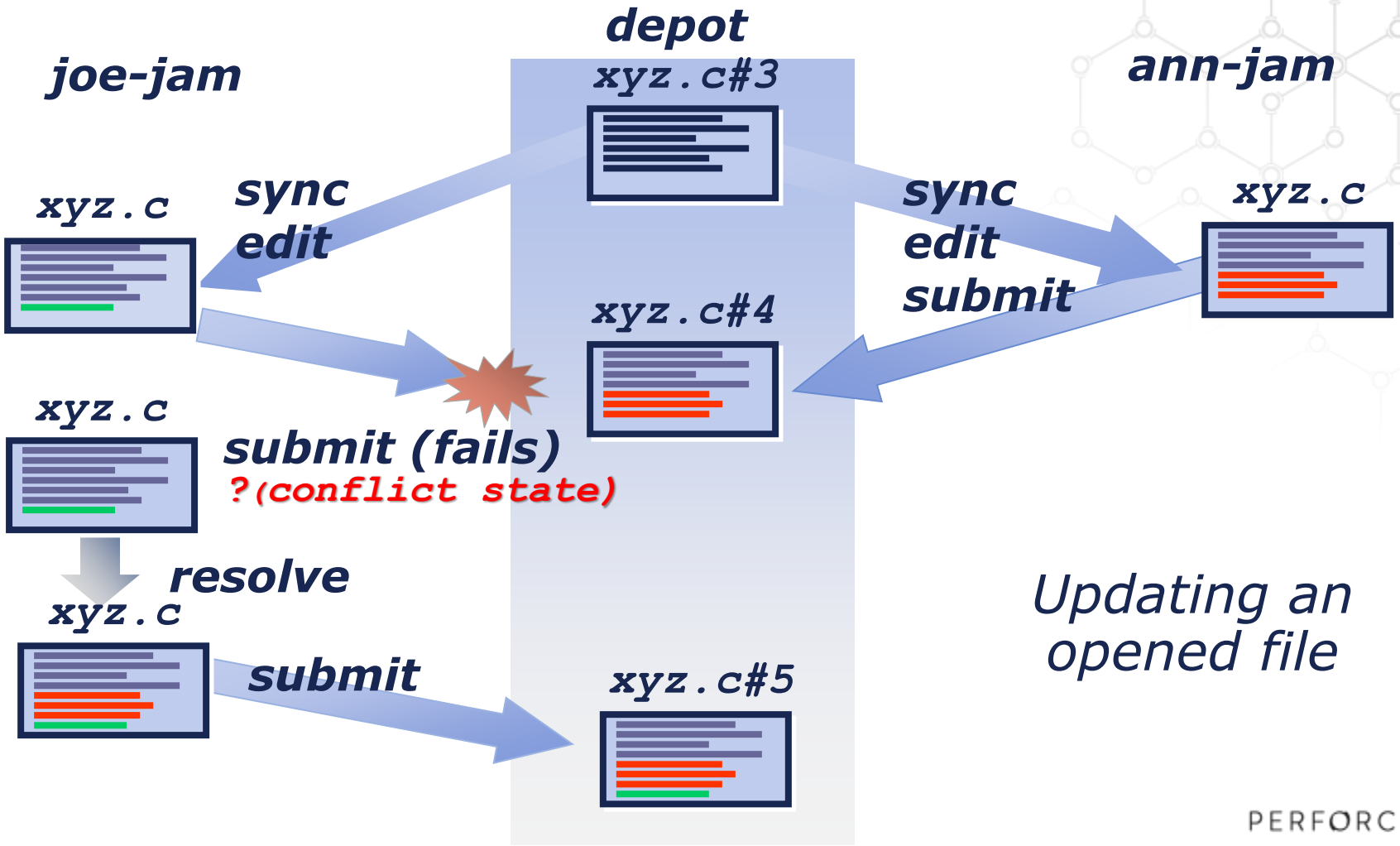
PERFORCE

Handling File Conflicts

- When can conflicts occur?
- Automatically resolve
- Interactively resolve
- Reporting Commands
- Backing out a Changelist

When Can Conflicts Occur?

- File conflicts
  - Submitting a change to an out-of-date file
  - Re-syncing an opened file that is out-of-date
  - Moving to an existing file
  - Unshelving onto an opened file
- Merge/resolve between branches/streams



Updating an opened file

When Can Conflicts Occur?

- Submitting changes to an out-of-date file

```
p4 submit
Change 5085 created with 4 open file(s).
Submitting change 5085.
//depot/Jam/MAIN/src/rules.c - must resolve before submitting
//depot/Jam/MAIN/src/rules.c - must resolve #6,#7
Out of date files must be resolved or reverted.
Submit failed -- fix problems above then use 'p4 submit -c 5085'
```

When Can Conflicts Occur?

- Re-syncing opened files

```
p4 sync -q ...
... //depot/Jam/MAIN/src/rules.c - must resolve #2,#5 before submitting
... //depot/Jam/MAIN/src/rules.h - must resolve #2,#14 before submitting
```

When Can Conflicts Occur?

- Moving to an existing file

```
p4 move -f //depot/Jam/REL2.1/... //depot/Jam/MAIN/...

//depot/Jam/MAIN/src/regexp.c#2 - moved from
                                  //depot/Jam/REL2.1/src/regexp.c#1
//depot/Jam/MAIN/src/regexp.h#1 - moved from
                                  //depot/Jam/REL2.1/src/regexp.h#1
```

When Can Conflicts Occur?

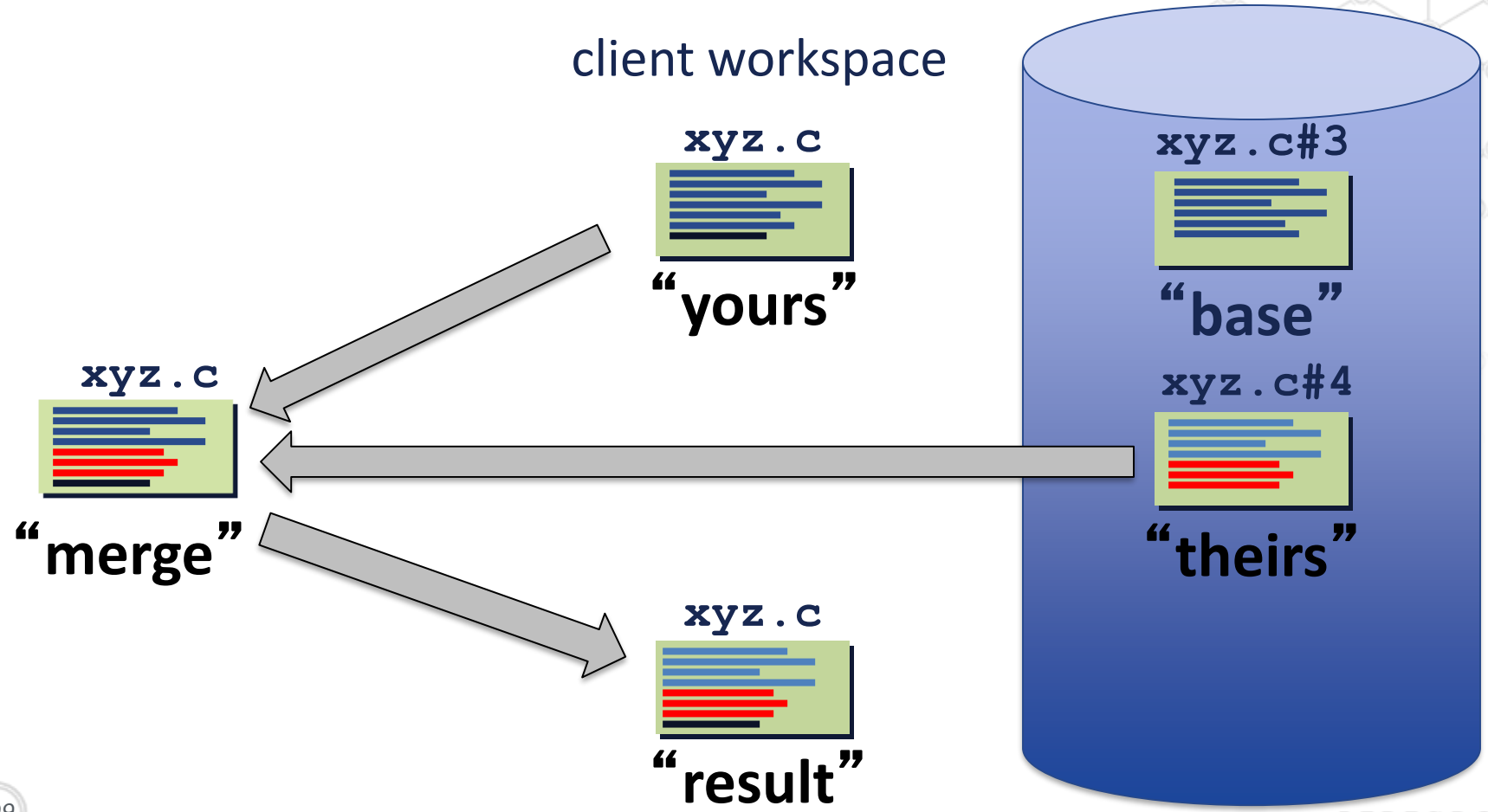
- Unshelving onto opened file

```
p4 unshelve -s 12023
```

```
//depot/Jam/MAIN/src/rules.c#5 - unshelved, opened for edit
... //depot/Jam/MAIN/src/rules.c - must resolve
  //depot/Jam/MAIN/src/rules.c@=12023 before submitting
```

What happens on conflict?

- Depot file remains unchanged
- A resolve is scheduled (needs to be performed)

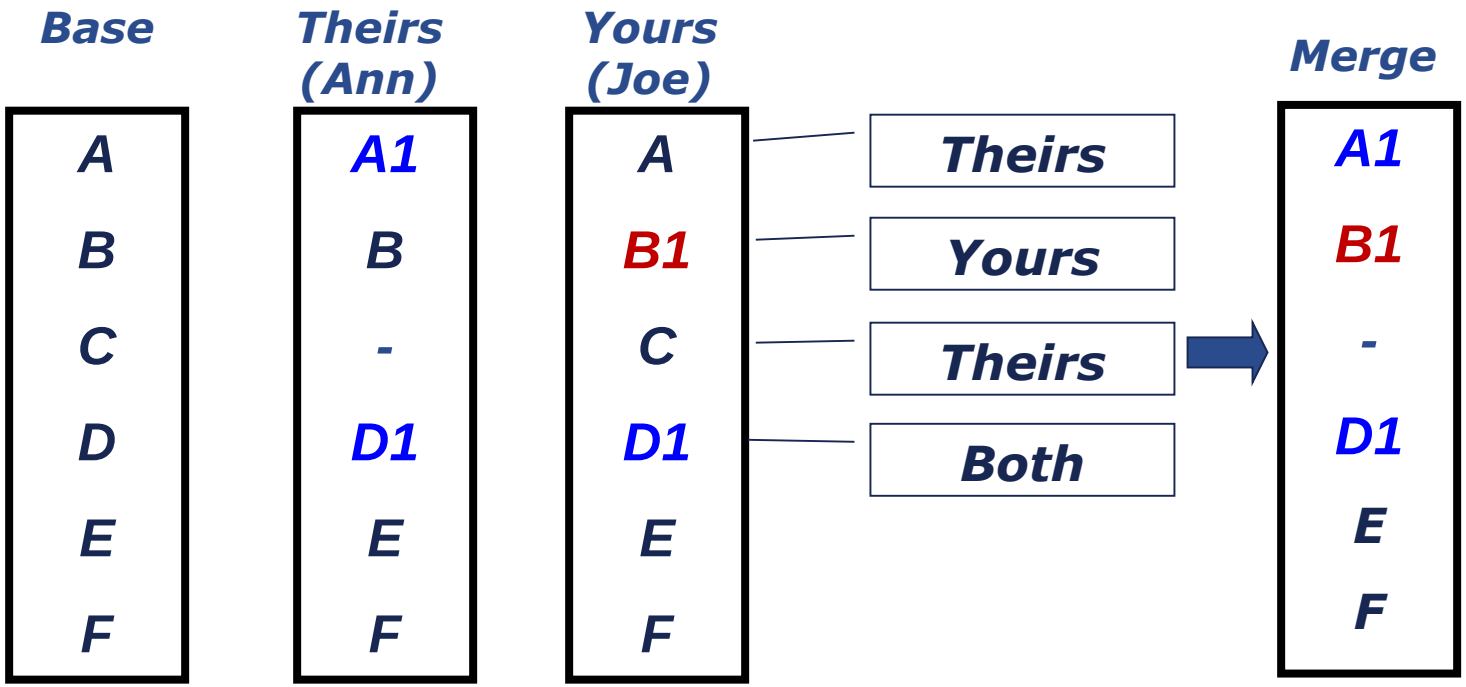

client workspace
xyz.c
“yours”
xyz.c#3
“base”
xyz.c
“merge”
xyz.c#4
“theirs”
xyz.c
“result”

Diff chunk types

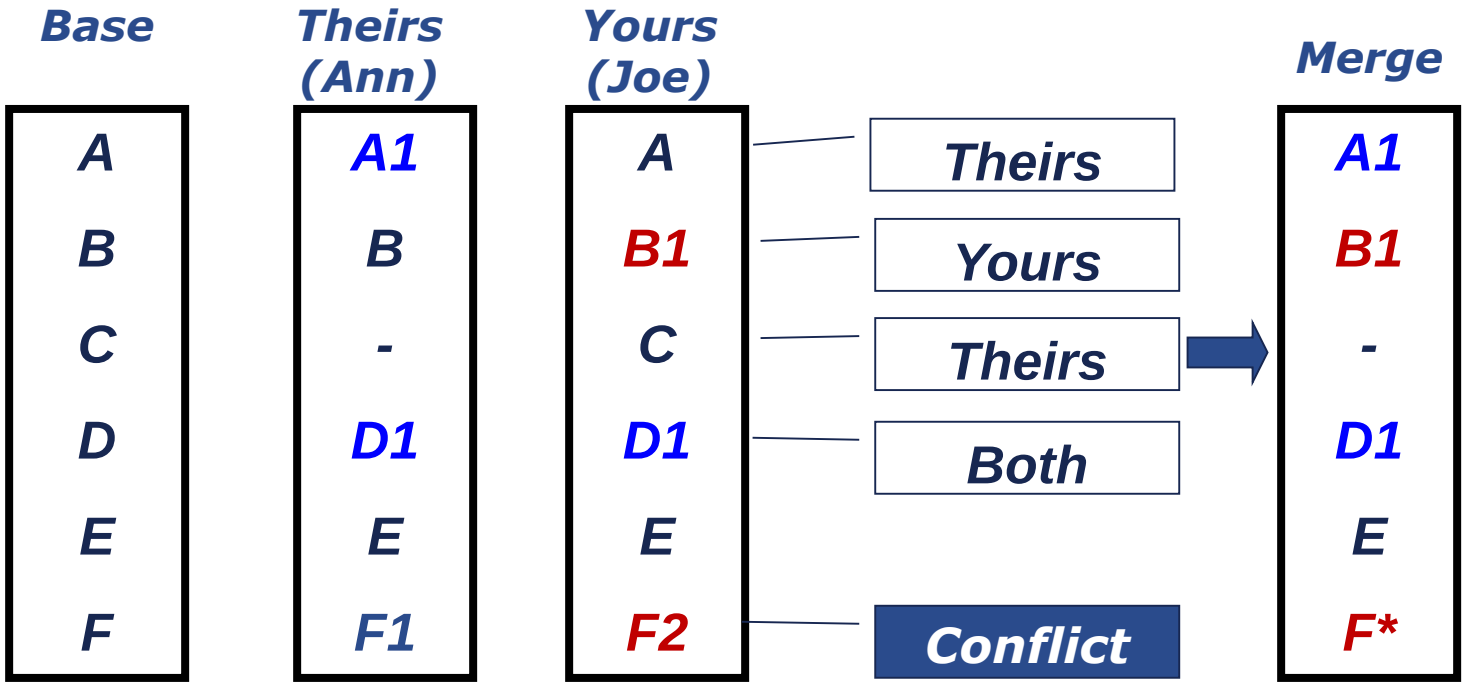
Unique diff chunk…

yours	Different workspace file only
theirs	Different depot file only
both	Common to depot and workspace file
conflicting	Conflicting between depot, workspace and base

Diff Chunks – Easy 3-way Merge



Diff Chunks – 3-way Merge conflicts



Automatic Resolve Options

- Ignore (accept your workspace file)

```
p4 resolve -ay
```

- Copy (accept their depot file)

```
p4 resolve -at
```

- Safe (accept yours or theirs, if only one differs)

```
p4 resolve -as
```

- Merge (accept yours, theirs or merge, if no conflicts)

```
p4 resolve -am
```

Interactive Resolve

```
p4 resolve
c:\work\Jam\MAIN\src\rules.c
  - merging //depot/Jam/MAIN/src/rules.c#6,#7
Diff chunks:
  0 yours + 1 theirs + 0 both + 1 conflicting
Accept(a) Edit(e) Diff(d) Merge (m)
Skip(s) Help(?) e:
```

- Iterates over each file scheduled for resolve.

Interactive Resolve Actions

- Accept yours (`ay`)
  - Ignore ‘theirs’, leave workspace file as-is
- Accept theirs (`at`)
  - Copy ‘theirs’ to workspace
- Accept merged (`am`)
  - Replace workspace file with ‘merge’ file
- Accept edited (`ae`)
  - Replace workspace file with edited ‘merge’ file

Before Resolving, You Can…

- **Edit(e)**
  - merged file
  - your file
- **Skip(s)**
- **Help(?)**

- **Diff(d)**
  - merged vs. yours
  - yours vs. base **(*dy*)**
  - theirs vs. base **(*dt*)**
  - merged vs. base（***dm***）

Editing the merged file (with conflicts)

theirs

```
# include "jam.h"
# include "option.h"
# include "patchlev.h"
# include "make.h"
```

yours

```
# include "jam.h"
# include "option.h"
# include "patchlevel.h"
# include "make.h"
```

merged

```
# include "jam.h"
# include "option.h"
>>>> ORIGINAL //depot/Jam/MAIN/xyz.c#3
==== THEIRS //depot/Jam/MAIN/xyz.c#4
# include "patchlev.h"
==== YOURS //bruno_ws/Jam/MAIN/xyz.c
# include "patchlevel.h"
<<<<
# include "make.h"
```

result (after editing)

```
# include "jam.h"
# include "option.h"
# ifdef FATFS
# include "patchlev.h"
# else
# include "patchlevel.h"
# endif
# include "make.h"
```

Textual vs. Binary Resolves

- Three-way resolves
  - Operate on text files
  - Use Yours, Theirs, and Base
  - Files can be merged
- Two-way resolves
  - Operate (by default) on binary files
  - No binary diffs stored, so files can't be merged
  - Resolve choices: accept yours or theirs

Resolve Reporting Commands

- List pending resolves

```
p4 resolve -n
c:\work\Jam\MAIN\src\rules.c -
        merging //depot/Jam/MAIN/src/rules.c#6,#7
```

- List resolved files

```
p4 resolved
c:\work\Jam\MAIN\src\hash.c -
        merged from //depot/Jam/MAIN/src/hash.c#4,#6
```

- Re-do a resolve

```
p4 resolve -f
```

Locking a File

```
p4 lock rules.c
//depot/Jam/MAIN/src/rules.c - locking
```

- Why?
  - Prevent others submitting
  - Avoid multiple resolves
- How is a file unlocked?
  - Revert
  - Submit
  - `p4 unlock <file>`

Backing Out a Changelist

```
p4 sync @6024

p4 edit index.c command.h

p4 sync

p4 resolve -ay

p4 submit
Change 6026 created with 2 open file(s).
Locking 2 files ...
Submitting change 6026.
edit //depot/Jam/MAIN/src/index.c#11
edit //depot/Jam/MAIN/src/command.h#6
Change 6026 submitted.
```

New Commands in This Chapter

- **`p4 resolve`**
- **`p4 resolved`**
- **`p4 lock`**

Introduction to Helix for Users

Workspace Management

PERFORCE

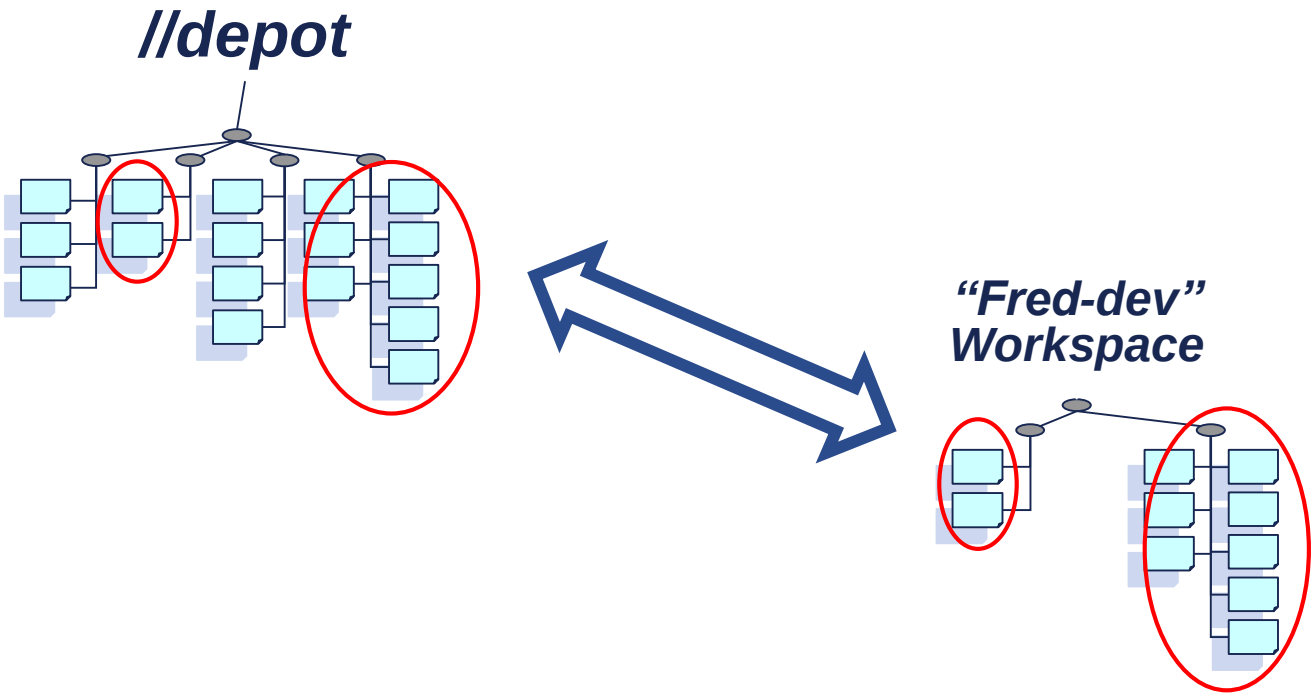
Workspace Management

- What is a workspace?
- Why have multiple workspaces?
- What does a workspace need?
- Creating a Workspace
- Workspace View Mappings

What is a Workspace?

- View into a Perforce Server
- Local files in the workspace are managed by Perforce
- Uniquely identified by its name (`P4CLIENT`)

Workspace Views map between server and local file system



Why Have Multiple Workspaces?

- Project/Branch oriented
- Better performance
  - For both you and your colleagues!

What Does a Workspace Need?

- **Name**
  - Uniquely identifies the workspace
- **Root**
  - The workspace's root directory on your client machine
- **View**
  - Maps areas of a Perforce Server to client workspace

Creating a Workspace

p4 client

```
Client:   bob-dev
Update:   2009/11/15 12:39:47
Access:   2010/06/03 16:33:25
Owner:    bob
Host:     perseus
Description:
          Bob's main development workspace.
Root:     c:\p4work
Options:  noallwrite noclobber nocompress unlocked
          nomodtime normdir
SubmitOptions: submitunchanged
LineEnd:  local
View:
  //depot/Jam/MAIN/... //bob-dev/Jam/MAIN/...
```

Client workspace view

```
Client: bob-jam
Root: c:\p4work\bob-jam
View:
  //depot/Jam/MAIN/src/...          //bob-jam/Jam/MAIN/src/...
 -//depot/Jam/MAIN/src/tests/...    //bob-jam/Jam/MAIN/src/tests/...
  //depot/Jam/MAIN/src/*.h          //bob-jam/Jam/MAIN/src/hfiles/*.h
  //depot/Jam/MAIN/A/B/config/*     //bob-jam/Jam/MAIN/cfg/*
  //depot/Acme-api/....cpp          //bob-jam/Acme-api/....CC
  //depot/Jam/REL1.0/src/...        //bob-jam/Jam/REL1.0/src/...
 +//depot/Jam/PATCH1.0/src/...      //bob-jam/Jam/REL1.0/src/...
```

Client Workspace Options

- Options

  `[no]allwrite` `[no]compress`
  `[no]clobber` `[no]locked`
  `[no]modtime` `[no]rmdir`

- SubmitOptions

  `submitunchanged``[+reopen]`
  `revertunchanged[+reopen]`
  `leaveunchanged[+reopen]`

- LineEnd

  `local` `unix` `mac` `win` `share`

Creating Workspaces

- Creating a new workspaces

```
p4 client bob-jam-project_a
```

- Copy a workspace's view and options

```
p4 client -t bob-jam-dev mary-jam-dev
```

- List all workspaces for a user

```
p4 clients -u bob
```

Testing Workspace View

- Show depot and client workspace paths

```
p4 where command.c
//depot/Jam/MAIN/src/command.c
//mel-dev/Jam/MAIN/src/command.c
d:\mel-dev\Jam\MAIN\src\command.c
```

- File does not have to exist for **`p4 where`** to work

New Commands in This Chapter

- **`p4 client`**
- **`p4 where`**

Introduction to Helix for Users

Intermission!

PERFORCE

Introduction to Helix for Users

Workspace Management (continued)

PERFORCE

Workspace Management

- Configuring Perforce Settings
- Setting up your client machine
- Checking your client configuration

Environment Variables

- Important for client applications:
  - `P4USER` – Perforce User name
  - `P4CLIENT` – Perforce workspace name
  - `P4PORT` – Perforce Server hostname and port
  - `P4CONFIG` – Client config filename

Setting in the Command

- Command syntax

  p4 *(usage flags)* **command** [flags] args

- *-p <port> -u <user> -c <workspace>*

  p4 *-p mars:1666 -u otto -c otto-api* **client**

  p4 *-c reb-jam-main* **sync** *-n*

Environment Setup

- Mac/Linux/UNIX (probably in .profile/.bashrc/…)

```
export P4PORT=neptune:1666
export P4CLIENT=lisas-jam-dev
export P4USER=lisa
```

- Windows/Mac

```
set P4PORT=neptune:1666
set P4CLIENT=lisas-jam-dev
set P4USER=lisa
```

Persisting Environment

- All Platforms

```
p4 set P4PORT=neptune:1666
p4 set P4CLIENT=lisas-jam-dev
p4 set P4USER=lisa
```

- Windows Only

```
p4 set -s P4CONFIG=.p4config
```

P4CONFIG File Setup

- Set **P4CONFIG** to a file name

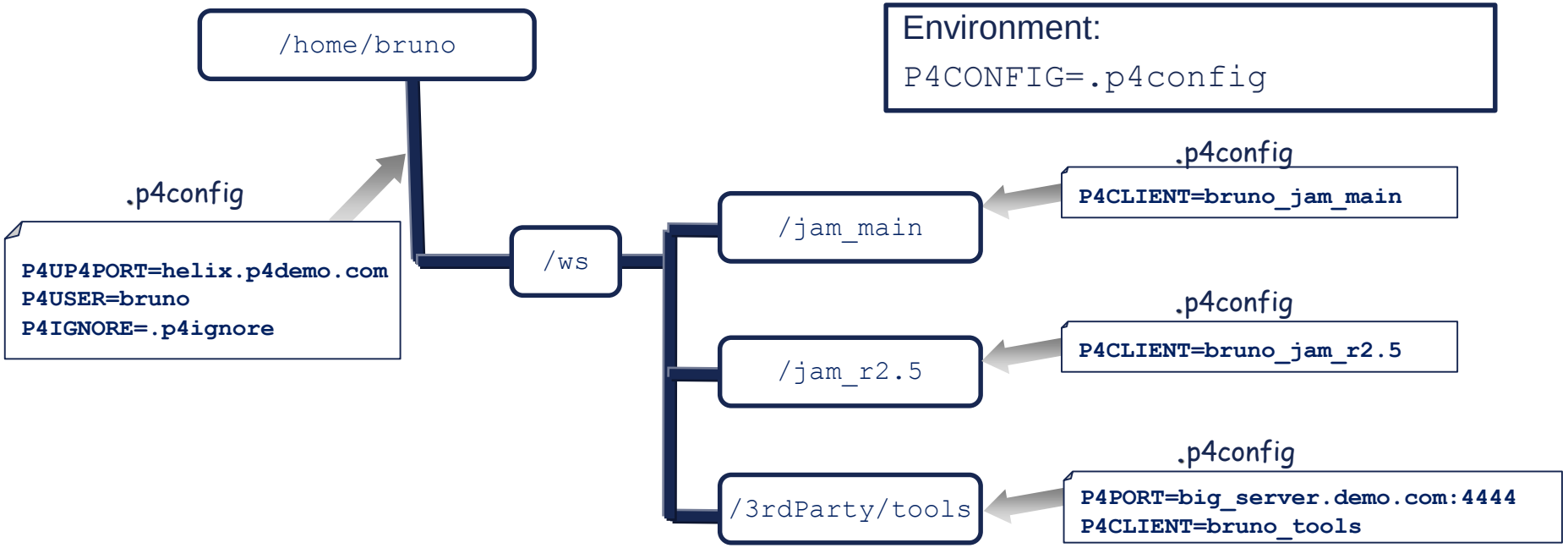
```
p4 set P4CONFIG=.p4config
```

- Create P4CONFIG file(s) in client workspace root

- List environment variables in named config file

```
P4PORT=lemon:1909
P4CLIENT=graphics_proj
```

P4CONFIG Sample Usage



Order of Precedence

1) Command-line flags
2) Variables defined in the **P4ENVIRO** file (non-Windows)
3) Variables defined in **P4CONFIG** file(s)
4) Environment variable settings
5) Registry variable settings (Windows only)
6) Default values for P4USER, P4PORT, P4CLIENT

Recommended Client Setup

- General settings in environment/registry
- Workspace-specific settings in **`P4CONFIG`** files

Checking Your Settings

- Use **`p4 set`** to list your configuration:

```
$ p4 set
P4CLIENT=bruno_ws
P4CONFIG=.p4 (set) (config 'noconfig')
P4PORT=1666
P4USER=bruno
```

Setting a Password

- Use **`p4 passwd`** to set password on server
- A ‘strong’ password…
  - …is at least eight characters
  - …includes two of:
    - Upper case characters
    - Lower case characters
    - Non-alphabetic characters

Logging into Perforce

- Start a session

```
p4 login
Enter password:
User marta logged in.
```

- End a session

```
p4 logout
User marta logged out.
```

Other Environment Variables

- **P4CHARSET, P4DIFF, P4MERGE**
- **P4EDITOR, P4TICKETS**
- **P4IGNORE**
- **TMP, TEMP**

New Commands in this Chapter

- **`p4 set`**
- **`p4 passwd`**
- **`p4 login`**
- **`p4 logout`**

Introduction to Helix for Users

Branching and Merging

PERFORCE

Branching and Merging

- Terminology
- Why branch?
- Creating a New Branch
- Propagating Changes across Brances

Terminology

- Branch
  - (noun) A set of related files (a.k.a. codeline) created as copy of its parent
  - (noun) A branch spec (branch mapping P4V)
  - (verb) To create a new branch

- Merge
  - To propagate changes between existing branches/streams (**`merge/copy`**)

Why Branch?

- Conflicting policies
- Task oriented development
- Discussed in more detail in:

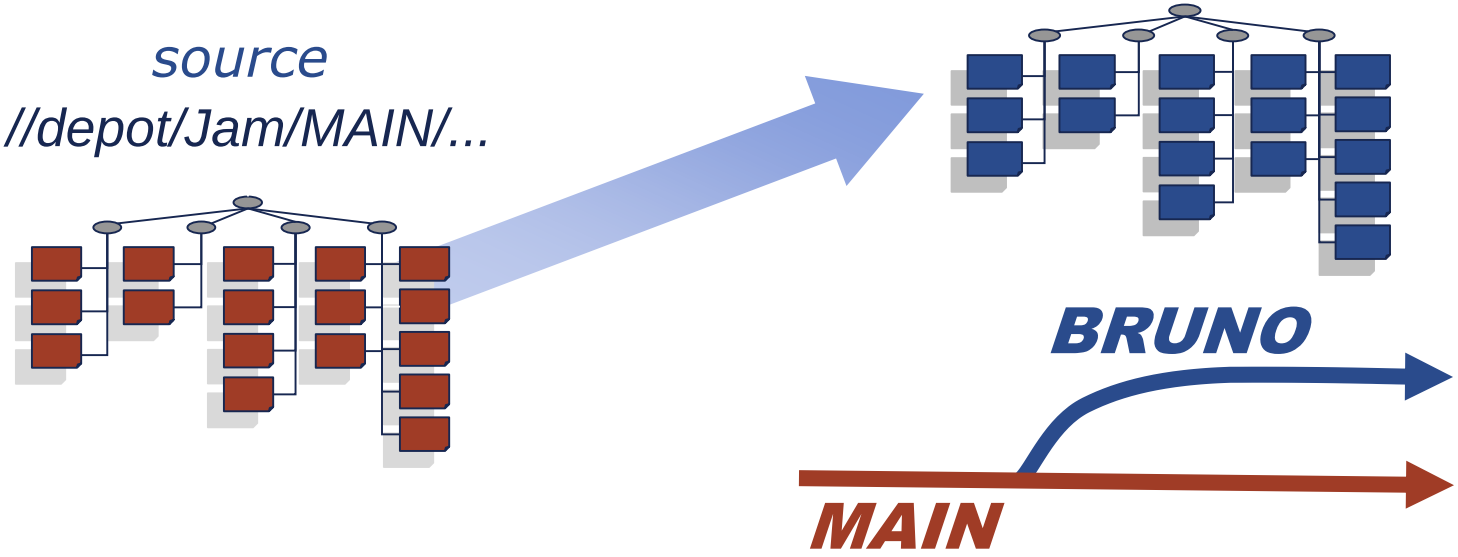
  High-Level SCM Best Practices

  http://www.perforce.com/sites/default/files/pdf/perforce-best-practices.pdf

Creating a branch

target
//depot/Jam/BRUNO/...

source
//depot/Jam/MAIN/...



How to Populate a Branch

- The source and target can be set in
  - A branch spec
    - Stores source/target pairs
    - Creates new branch
    - Re-use to propagate changes
  - A file spec
    - Source/target set on the command line
    - Once use
    - Useful for simple views.
  - A stream spec
    - More Later …

Creating Branch Spec

- Create a branch spec

```
p4 branch JamMAIN-to-R1.0
```

```
Branch:  JamMAIN-to-R1.0
Owner:   bob
Description:
      Jam Release 1.0 branch for Server & API
Options: unlocked
View:
 //depot/Jam/MAIN/svr/...   //depot/Jam/R1.0/svr/...
 //depot/Jam/MAIN/api/...   //depot/Jam/R1.0/api/...
```

source paths

target paths

Using populate

- Open files for integration
  - Branch spec

```
p4 populate -b JamMAIN-to-R1.0
```

  - File spec

```
p4 populate //depot/Jam/MAIN/... //depot/Jam/R1.0/...

71 files branched (change 12109).
```

Integration Steps – Adjust View

- Add target branch to workspace view

```
p4 client
```

```
Client: ben-jam
Description:
   Ben's Jam client workspace.
Root:   c:\demo
View:
   //depot/Jam/MAIN/...    //ben-jam/Jam/MAIN/...
   //depot/Jam/R1.0/...    //ben-jam/Jam/R1.0/...
```

Using p4 populate

- Create a new branch:
  - Branch spec

```
p4 populate -b JamMAIN-to-R1.0
```

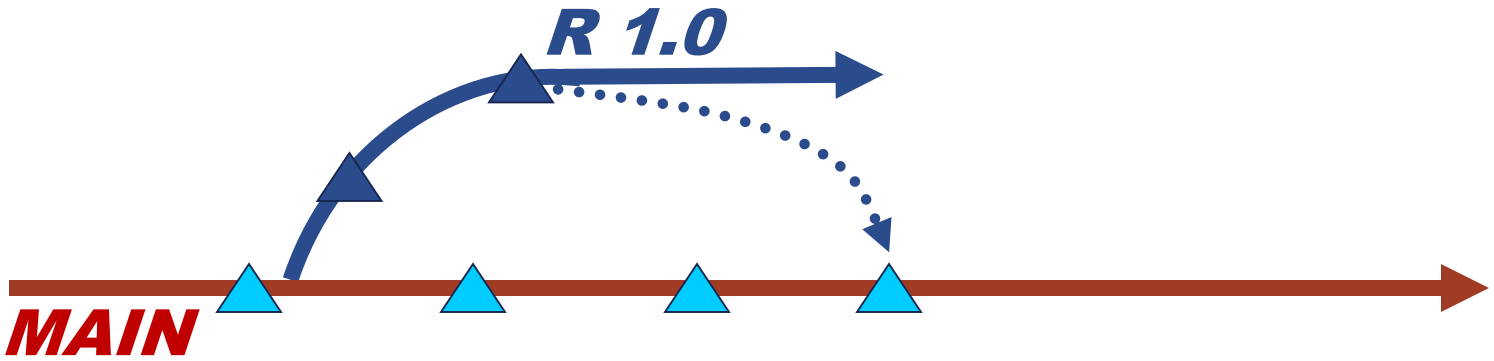
  - File spec

```
p4 populate //depot/Jam/MAIN/... //depot/Jam/R1.0/...
```

  - Stream spec (covered later)

```
p4 populate -S //jam/R1
```

Propagating changes across branches



Integration steps – propagating changes

- Check Workspace View
- Open files for integration
- Resolve
- Submit

Integration steps – propagating

- Open files for integration
  - Branch spec
    - `p4 merge -r -b JamMAIN-to-R1.0`
  - File spec
    - `p4 merge //depot/Jam/R1.0/... //depot/Jam/MAIN/...`
- Perforce tracks integration history
- Only changed files are propagated

Integration steps – propagating

- Resolve

```
p4 resolve
```

- Submit

```
p4 submit
```

Branched revision history

```
p4 filelog -i //depot/Jam/REL2.1/src/hash.c

//depot/Jam/REL2.1/src/hash.c
... #2 change 869 edit ...
... #1 change 749 branch ...
... ... branch from //depot/Jam/MAIN/src/hash.c#1,#3
//depot/Jam/MAIN/src/hash.c
... #3 change 137 edit ...
... ... branch into //depot/Jam/REL2.1/src/hash.c#1
... #2 change 30 edit ...
... ... branch into //depot/Jam/REL2.0/src/hash.c#1
... #1 change 1 add ...
```

- P4V's Revision Graph gives a graphical view

New Commands in this Chapter

- `p4 branch`
- `p4 merge`
- `p4 populate`
- `p4 filelog -i`

Introduction to Helix for Users

Branching and Merging (continued)

PERFORCE

Branching and Merging

- Different resolve types
- Integration reporting
- The Flow of Change

Different resolve types

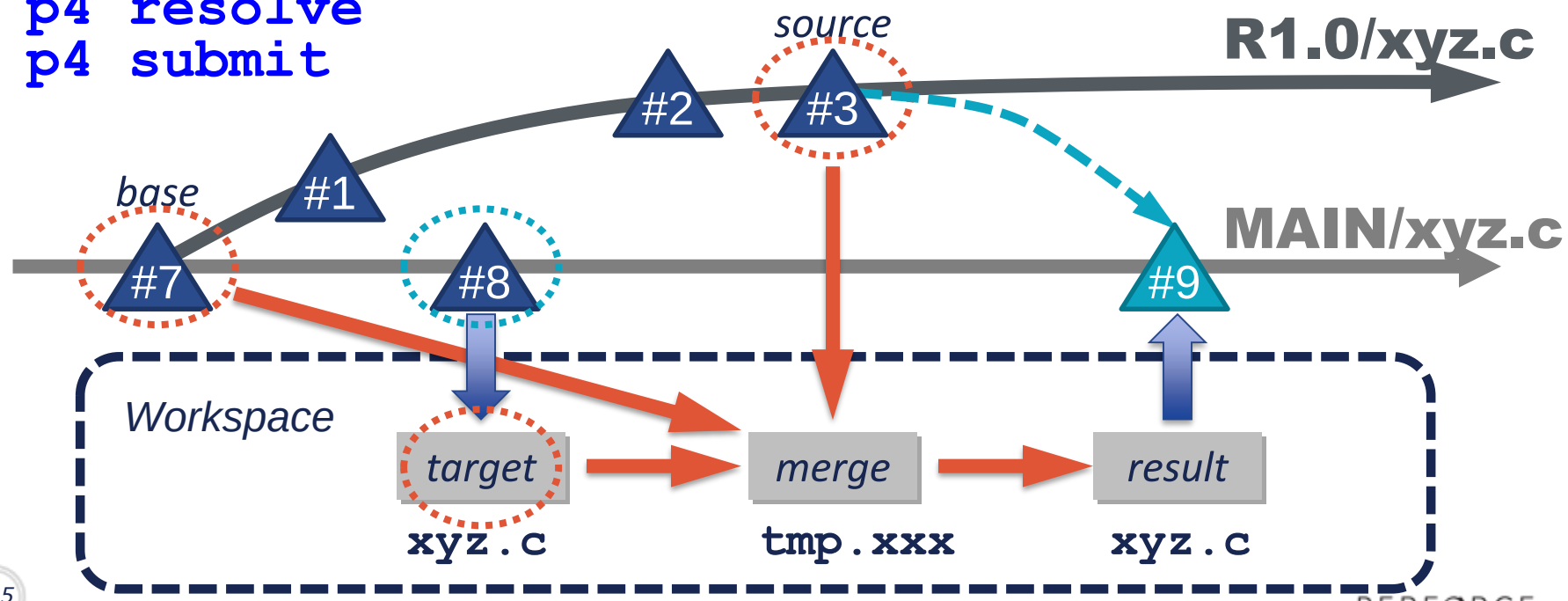
- Content
- Moved files
- Filetype changes
- Branched files
- Deleted files

Resolving for merge/integrate

- *Theirs* = Source
- ***Yours*** **= Target (the one being changed by merge)**
- *Base* = Closest common ancestor

Resolves during merge

```
p4 merge -r -b JamMAIN-to-R1.0
p4 resolve
p4 submit
```



Moved files

- Source or target renamed using **p4 move**
  - Copy source file name (**at**)
  - Ignore source file name (**ay**)

p4 resolve

```
c:\work\Jam\DevA\src\fileapple.c -
    resolving move to //Jam/DevA/src/fileosx.c
Filename resolve:
at: //Jam/DevA/src/fileosx.c
ay: //Jam/DevA/src/fileapple.c
Accept(a) Skip(s) Help(?) at:
```

Filetype changes

- Filetype changed in source, target or both
  - Copy source file type (**at**)
  - Ignore source file type (**ay**)
  - Merge file types (**am**)

```
p4 resolve
c:\work\Jam\DevA\src\seek.c - resolving filetype from
    //Jam/MAIN/src/seek.c#2
Filetype resolve:
at: (binary+m)
ay: (binary+l)
am: (binary+lm)
Accept(a) Skip(s) Help(?) am:
```

Branched Files

- Integrate branches new files, by default
- Schedule branch resolve with **p4 integ -Rb**
  - Branch file (**at**)
  - Ignore file (**ay**)

```
p4 resolve
c:\work\Jam\DevA\src\newfile.txt - resolving branch from
    //Jam/MAIN/src/newfile.txt#1
Branch resolve:
at: branch
ay: ignore
Accept(a) Skip(s) Help(?) at:
```

Deleted Files

- Replaces the need for `-d` and `-D[sti]` flags
- Schedule delete resolve with **`p4 integ -Rd`**
  - Delete file (`at`)
  - Ignore delete (`ay`)

```
p4 resolve
c:\work\Jam\DevA\src\search.c - resolving delete from
   //Jam/MAIN/src/search.c#3
Delete resolve:
at: delete
ay: ignore
Accept(a) Skip(s) Help(?) at:
```

What needs to be integrated?

```
p4 interchanges -r -b JamREL2.1toMAIN
Change 183 on 2001/02/06 by earl@earl-dev-guava 'Straightened out platform su'
Change 185 on 2001/02/20 by earl@earl-os2-buckeye 'Rework Jambase to confor'
Change 186 on 2001/02/23 by earl@tosh 'Have jam's makeDirName return'
Change 187 on 2001/03/05 by earl@earl-dev-guava 'New LOCATE_SOURCE for jam.'
```

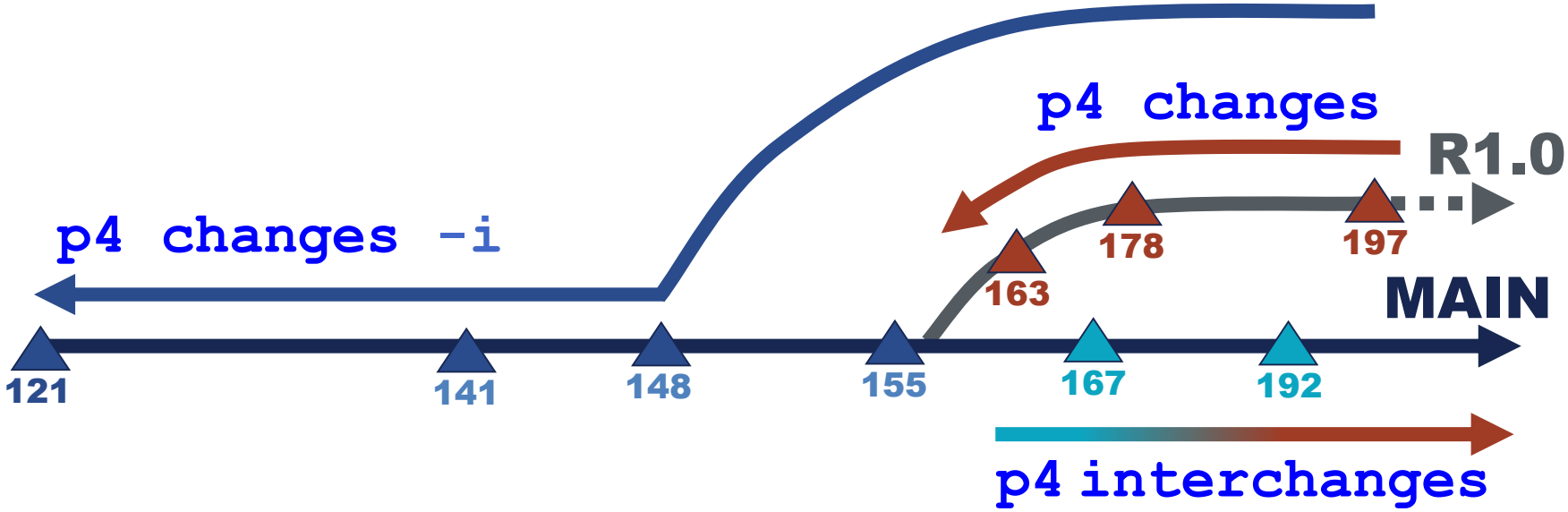
What has been integrated?

```
p4 integrated //depot/Jam/REL2.1/...
//depot/Jam/REL2.1/src/Build.com#1 - branch from
                                   //depot/Jam/MAIN/src/Build.com#1,#5
//depot/Jam/REL2.1/src/Build.com#1 - branch into
                                   //Jam/REL2.1/src/Build.com#1
//depot/Jam/REL2.1/src/command.c#1 - branch from
                                   //depot/Jam/MAIN/src/command.c#1,#5
//depot/Jam/REL2.1/src/command.c#1 - branch into
                                   //Jam/REL2.1/src/command.c#1
...etc...
```

What changes were integrated?

```
p4 changes -i //depot/Jam/R1.0/...
Change 5047 on 2008/12/29 by bob@bob-jam 'Create release branch'
Change 5028 on 2008/12/23 by baron@baron-bios 'Upgrade to latest metrowerks'
Change 5026 on 2008/12/22 by jean@jean-spice 'Added all the old press release'
Change 5025 on 2008/12/22 by jean@jean-spice 'Press releases now just use sta'
Change 5014 on 2008/12/19 by berger@berger-spice 'Put in fix for jam's NT han'
...etc...
```

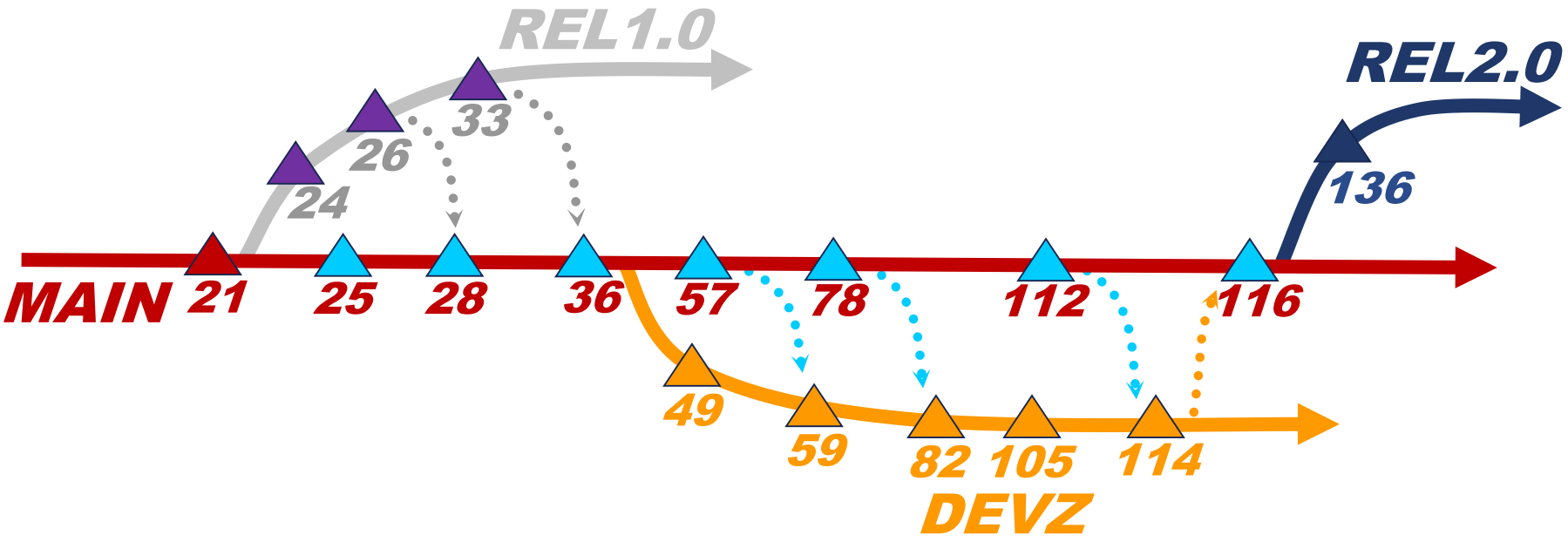
What changes are reported?



What differs between branches?

```
p4 diff2 -q -b MAIN-to-R1.0
 //depot/Jam/MAIN/src/command.c#9      (text) -
    //depot/Jam/R1.0/src/command.c#1   (text) content
 //depot/Jam/MAIN/src/regexp.c#2       (text) -
    //depot/Jam/R1.0/src/regexp.c#2    (text) content
 //depot/Jam/MAIN/src/RELNOTES#77      (text) -
    //depot/Jam/R1.0/src/RELNOTES#2    (text) content
 //depot/Jam/MAIN/src/scan.c#18 -      <none>
 //depot/Jam/MAIN/src/scan.h#12 -      <none>
<none> - //depot/Jam/R1.0/src/truescan.h#1
 ...etc...
```

Some branching patterns



A Mantra: Merge Down, Copy Up

`p4 merge`

- Merge from source to target
- Simplified form of legacy **`p4 integ`** command
- Files need to be resolved

`p4 copy`

- Simple copy from source to target
- Ignores previous integrations
- No manua resolve needed.

New Commands in this Chapter

- **`p4 interchanges`**
- **`p4 integrated`**
- **`p4 copy`**
- **`p4 merge (p4 integ)`**

Introduction to Helix for Users

Streams

PERFORCE

Streams

- Introduction
- Creating a stream
- Populating a stream
- Working in a stream
- Streams views

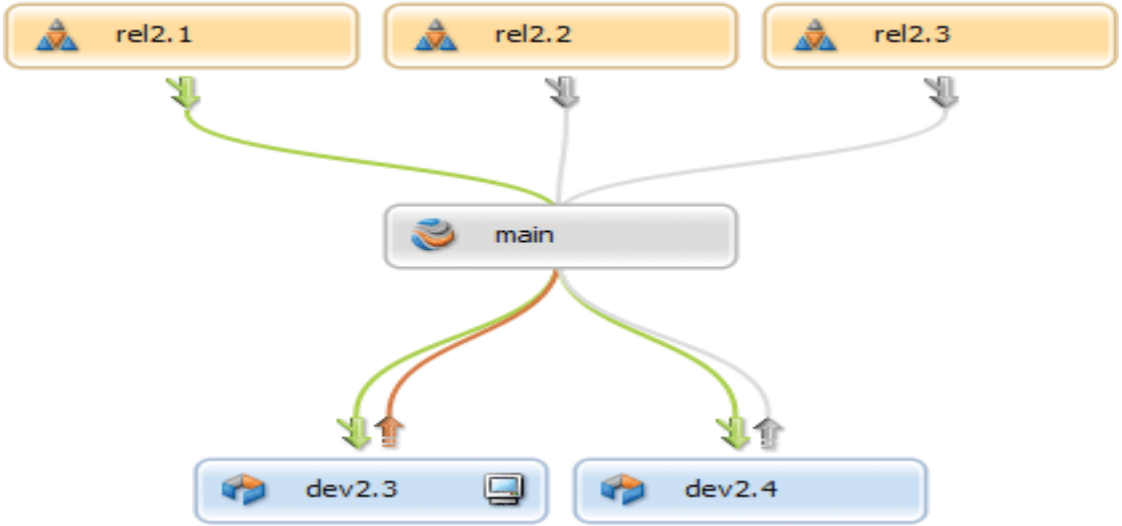
Introduction

- A Stream is a “Branch with Brains”
- Workflow defined visually
- Guides users to follow a flow of change
- Simplifies merge mechanics
- 1:1 relationship
- Based on the mainline model
- Defined relationship to its parent...

  ...not to its children

Stream relations

- Controlled flow of change



Stream Types

‘release’

Highly stable

(merge ‘down’ to parent, copy ‘up’ from parent)

‘mainline’

Stable per your policy

(merge ‘down’ to dev, copy ‘up’ from dev)

(merge ‘down’ from release, copy ‘up’ to release)

‘development’

Unstable

(merge ‘down’ from parent, copy ‘up’ to parent)

Creating a mainline Stream

- Requires a streams depot (e.g. `//projX`)
- Create a ‘mainline’ stream spec

```
p4 stream -t mainline //projX/main
```

Creating a Streams Workspace

- Creating a Streams workspace
  - Run **`p4 client`** as normal
  - Add stream field and name:

    `Stream: //projX/main`
  - The **`View:`** field is automatically generated
    - user updates are ignored/not saved

- Switching to another stream (updates local files)

  `p4 client -s -S //projA/dev`

Populating the mainline

- Add files from local file system

```
dir /s/b/a-d  |  p4 -x - add
p4 reconcile
```

- Or branch from another depot (local or stream depot)

```
p4 copy -v //src/... //projX/main/...
```

Editing a Stream Spec

```
p4 stream //projX/main
```

```
Stream:       //projX/main
Update:       2011/12/07 17:51:56
Access:       2012/01/04 14:41:06
Owner:        bruno
Name:         main
Parent:       none
Type:         mainline
Options:      allsubmit/ownersubmit
                  [un]locked   [no]toparent
                  [no]fromparent
```

Creating a Child Stream

- Creating a ‘development’ stream

```
p4 stream -t development -P  //projX/main //projX/dev
```

- Creating a ‘release’ stream

```
p4 stream -t release -P  //projX/main //projX/rel
```

Populating streams

- Populate a child from its parent

```
p4 populate -S //new/dev -r
```

  `-S` maps child to parent

  `-r` reverse direction

- Populate from an unrelated parent

```
p4 populate -S //other/main -P //new/main
```

  `-P` target stream

Propagating Changes

- Propagate changes using:

```
p4 merge -S <stream>
p4 copy -S <stream>
p4 integ -S <stream>
```

- To merge down from ‘main’ to ‘dev’

```
p4 merge -S //projX/dev -r
```

- To copy up from ‘dev’ to ‘main’

```
p4 copy -S //projX/dev
```

Listing Streams

- List all streams

```
p4 streams
```

- Restrict list to a specific depot

```
p4 streams //projX/...
```

- Use '-F' to filter results

```
p4 streams -F "type=release" //projX/...
```

New Commands in this Chapter

- `p4 stream`
- `p4 streams`

Introduction to Helix for Users

Labels

PERFORCE

About Labels

- Labels vs. Changelist Numbers
- Creating a Label
- Tagging Files with a Label
- Static vs. Automatic Labels
- Locking a Label

Labels vs. Changelist numbers

- User determined file set
- Can be changed
- Meaningful name

Tagging files with a label (static)

```
p4 tag -l rel1 //depot/Jam/R1.0/...
//depot/Jam/R1.0/svr/jamfile#1 - added
//depot/Jam/R1.0/svr/libsvc.c#3 - added
//depot/Jam/R1.0/svr/libsvc.h#5 - added
//depot/Jam/R1.0/svr/rmtsvc.c#8 - added
//depot/Jam/R1.0/svr/rmtsvc.h#2 - added
//depot/Jam/R1.0/svr/userhelp.c#4 - added
...etc...
```

Updating a label

- Use **p4 tag**

```
p4 tag -l rel1 //depot/Jam/MAIN/...@123
```

- Adding/Updating references to files

```
p4 tag -l rel1 //depot/Jam/MAIN/...
```

- Delete references to files

```
p4 tag -d -l rel1 //depot/Jam/MAIN/svr/...
```

Automatic Labels

```
p4 label Main-build-Apr23_15
```

```
Label:   Main-build-Apr23_15
Owner:   bob
Description:
    Build label for development work.
Options:  unlocked noautoreload
Revision:     @30357
View:
  //depot/Jam/MAIN/svr/...
  //depot/Jam/MAIN/api/...
```

Unloaded Labels

```
p4 label Main-build-Apr23_15
```

```
Label:   Main-build-Apr23_15
Owner:   bob
Description:
    Build label for development work.
Options:  unlocked autoreload
View:
  //depot/Jam/MAIN/svr/...
  //depot/Jam/MAIN/api/...
```

```
p4 tag -l Main-build-Apr23_15 @10357
```

Using labels

- Syncing to it

```
p4 sync @rel1
p4 sync //depot/Jam/...@rel1
p4 sync @rel1,@rel1
```

- Listing files tagged by it

```
p4 files @rel1
p4 files //depot/Jam/...@rel1
```

Locking a Label

```
p4 label jam1.0
```

```
Label:    jam1.0
Owner:    bob
Description:
     Label for Release 1.0 - production
Options:  locked noautoreload
Revision:     @10357
View:
   //depot/Jam/REL1.0/...
```

Listing Labels

```
p4 labels -e "jam-*"
Label jam-1.1.0 2008/01/09 'Created by earl. '
Label jam-2.0.5 2009/01/01 'Created by earl. '
Label jam-2.1.0 2009/05/04 'Created by earl. '
...etc...
```

```
p4 labels //depot/Jam/MAIN/...
Label jam-1.1.0 2008/01/09 'Created by earl. '
Label jam-2.0.5 2009/01/01 'Created by earl. '
Label jam-2.1.0 2009/05/04 'Created by earl. '
...etc...
```
(*only static and not unloaded labels*)

New Commands in this Chapter

- `p4 tag`
- `p4 label`
- `p4 labels`

Introduction to Helix for Users

Job Tracking

PERFORCE

Job Tracking

- Perforce Jobs
- Creating a Job
- Searching Jobs
- Linking Jobs to Changelists

Perforce jobs

- Perforce’s defect tracking system
- Textual descriptions of unit of work
- Can be customized
- Can integrate with external defect tracker

Creating a job

```
p4 job
```

```
Job:      new

User:     bob

Status:   open

Date:       2009/08/01

Description:
     Add buttons to Gizmo space window.
```

Editing an existing job

```
p4 job job100123
```

```
Job:      job100123

User:     bobby

Status:   open

Date:       2014/08/01

Description:
     Add buttons to Gizmo space window.  Buttons should be
     mauve or taupe.
```

Listing jobs

p4 jobs

```
job000001 on 1999/01/19 by aban *closed* 'Add clean uninstall function'
job000002 on 1999/02/28 by aban *closed* 'Need release notes for jam'
job000003 on 1999/02/28 by leslie *closed* 'Make serialized release'
job000004 on 1999/07/19 by aban *closed* 'Jam port to FreeBSD. '
job000005 on 1999/10/23 by aban *closed* ‘Jam port to SCO. '
...etc...
```

Listing fixed jobs against a branch

```
p4 jobs //depot/Jam/MAIN/src/...
```

```
job011001 on 2014/01/23 by tsmith *closed* 'execunix.c blows up on'
job011162 on 2014/03/21 by tsmith *closed* 'MAXLINE on Win can't ac'
job014053 on 2015/01/22 by hjones *closed* 'Jam port to QNX'
```

Searching jobs

```
p4 jobs -e "filter email"

p4 jobs -e "status|email"

p4 jobs -e "status=open user=edk"

p4 jobs -e "date>=2014/07/14"

p4 jobs -e "description=filter*"

p4 jobs -e "doesn't return zero"
```

Setting a job view

p4 user

```
User:  bob
Email: bob@caniche.com
Update:       2008/08/09 13:45:59
Access:       2010/06/07 16:45:05
FullName:     Bob Everly
JobView:      user=bob status=open
```

Linking jobs to changelists

- Link changelist and job

  ```
  p4 fix -c changenum jobname
  ```

- Submit sets job status to `fixStatus`
  - Override using `p4 submit -s status`

Job fix reporting

- Which changelists fixed a job?

```
p4 fixes -j jobname
```

- Which jobs were fixed by a changelist?

```
p4 fixes -c changenum
```

- Same output for both

`p4 fixes -j job001234` or `p4 fixes -c 12345`

```
job001234 fixed by change 12345 on 2015/01/22 by bobby@esau
```

New Commands in this Chapter

- **`p4 job`**
- **`p4 jobs`**
- **`p4 user`**
- **`p4 fix`**
- **`p4 fixes`**

Introduction to Helix for Users

DVCS Quick Tour

PERFORCE

What is DVCS?

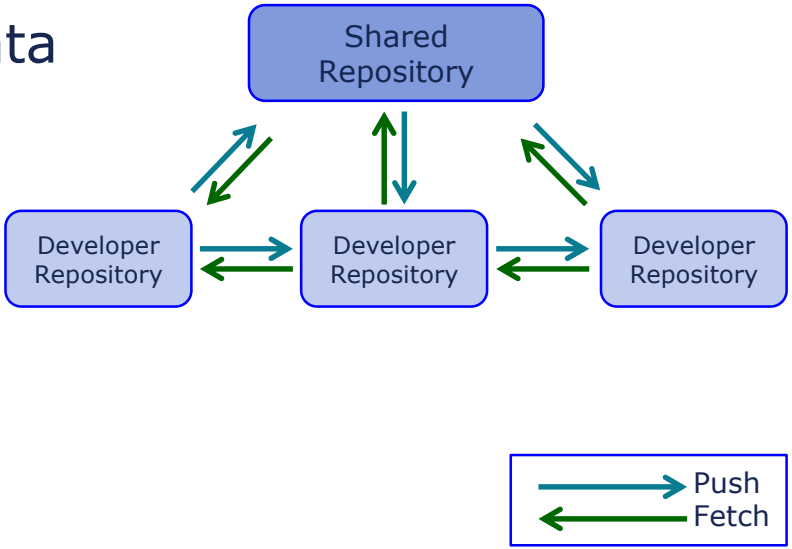
- **D**istributed
- **V**ersion
- **C**ontrol
- **S**ystem

What does DVCS offer?

- Local experimentation or development
- Offline commits (train, plane, behind firewalls, etc.)
- Immediately available local repository
- Wide variety of workflows available

DVCS architecture

- Push/fetch protocol
  - Fast & built for large data sets
- Easily move content and metadata
  - Between servers and peers
- Relocate content
  - Part of push/fetch process
- History and changes tracked
  - As with other Helix Server operations
- Security / access controls



Some DVCS workflows

- Solo work
  - User who does not want or need to share with anyone
- Work disconnected
  - Continue working with the system without direct access to master server
- Pull from master
  - Content is distributed to 3rd parties, groups, etc. from a central source
- Cross trust collaboration
  - Users never connect to the network but share data via zip files.

p4 init – Create personal repository

- **`p4 init`** - Creates a new personal repository
- Single command
- p4d executable must be in path
- Attempts to discover proper configuration
  - Case sensitivity
  - Unicode
- Copies no data from any server it discovers
- No p4d process runs!

p4 init – Create personal repository

- **p4 init** – with discovery:

```
$ p4 init
Matching server configuration from 'p4svr.perforce.com:1666':
case-sensitive (-C0), non-unicode (-n)
Server reb-dvcs-1431105951 saved.
```

- **p4 init** specifying case handling

```
$ p4 init -C0
Server reb-dvcs-1431108157 saved.
```

p4 info – results!

```
$ p4 info
User name: reb
Client name: reb-dvcs-1431108157
Client root: /home/reb/xyzzy
Client stream: //stream/main
Current directory: /home/reb/xyzzy
Peer address: unknown
Client address: unknown
Server address: personal server created by 'init'
Server root: /home/reb/xyzzy/.p4root
Server date: 2015/05/08 14:15:39 -0400 EDT
Server uptime: 00:00:00
Server version: P4D/LINUX26X86_64/2015.1/1028542 (2015/03/20)
ServerID: reb-dvcs-1431108157
Server license: none
Case Handling: sensitive
```

Use instantly

- Server environment set up and ready to use

```
$ echo "This is a test" > file.txt
$ p4 status
file.txt - reconcile to add //stream/main/file.txt#1
$ p4 reconcile
//stream/main/file.txt#1 - opened for add
$ p4 submit -d "Add our first file"
Submitting change 1.
Locking 1 files ...
add //stream/main/file.txt#1
Change 1 submitted.
$
```

Easily create and use additional branches

- **p4 switch**
  - Shows current branch
  - Easily create and switch between branches

```
$ p4 switch
main
$ echo "Hello world!" > helloworld.py
$ p4 switch -c dev
dev
$ p4 switch -l
dev *
Main
$
```

Easily create and use additional branches

- Lightweight branching takes care of your workspace
  - New file **helloworld.py** was created in main branch
  - Switch back to access that workspace

```
$ ls
file.txt
$ p4 switch main
$ ls
file.txt  helloworld.py
$
```

p4 init vs. p4 clone

- **`p4 init`**
  - Initialize a new personal (local) Helix Server
  - Server initialized with no data in it

- **`p4 clone`**
  - Create a new local Helix Server from a remote server
  - Desired portion of remote servers contents copied locally

Clone from existing repository

- **p4 clone**

```
$ mkdir talkhouse && cd talkhouse
$ p4 clone -p 1492 -f //depot/Talkhouse/main-dev/...
Server reb-dvcs-1431436869 saved.
Cloning from '1492'...
fetch load revisions 339 finishing
fetch load integrations 670 finishing
check files 100% finishing
fetch archives 100% finishing
commit revisions 100% finishing
sync 321 finishing
18 change(s) containing a total of 339 file revision(s) were …fetched.
$ p4 remotes
origin 1492 'auto-generated from clone command '
```

  - Use **p4 clone**'s **-m** flag to limit the number of revisions copied

Connecting to remote repos

- **`p4 remote`**
  - Describes the shared server your server cooperates with
    - Remote spec with
    - RemoteID – Identifier of the remnote
    - Address: – P4PORT use by the remote server
    - Lastfetch – Last changelist that was fetched
    - LastPush – Last changelist that was pushed
    - DepotMap: – Mappings between local and remote files
- **`p4 remotes`**
  - Displays a list of remote specifications

Remote spec

```
RemoteID:       origin
Address:        1492
Owner:  reb
Options:        unlocked nocompress
Update: 2015/05/12 06:25:02
Access: 2015/05/15 16:29:47
Description:
        auto-generated from clone command

LastFetch:      4554
LastPush:       877
DepotMap:
        //stream/main/... //depot/Talkhouse/main-dev/...
```

Make some changes… and push back

- Edit files, submit to local workspace, push!

```
$ p4 reconcile
//stream/main/build/build.xml#1 - opened for edit
//stream/main/build/build.properties#1 - opened for edit
$ p4 submit -d "Changed build scripts"
Submitting change 877.
Locking 2 files ...
edit //stream/main/build/build.properties#2
edit //stream/main/build/build.xml#2
Change 877 submitted.
$ p4 push
1 change(s) containing a total of 2 file revision(s) were successfully pushed
$ p4 -p 1492 changes -m 1
Change 12106 on 2015/05/08 by reb@reb-dvcs-1431437102 'Changed build scripts'
```

DVCS commands

- **`p4 init`** - Create a new personal server
- **`p4 clone`** - Clone a new personal server from shared server
- **`p4 remote`** - Define a connection to a shared server
- **`p4 fetch`** - Copy files from shared server to personal server
- **`p4 push`** - Copy files from personal server to shared server
- **`p4 switch`** - Switch to new stream, optionally creating it
- **`p4 unsubmit`** - Unsubmit a change, leaving the work in a shelf
- **`p4 resubmit`** - Resubmit unsubmitted changes

Finally...

Visit Perforce

http://www.perforce.com/

and the Perforce Workshop

https://workshop.perforce.com

for current listings of available software

Perforce Highlights

FAST

Easy to learn

Superior technical support

Cross-platform compatibility

Atomic change transactions

Powerful integration algorithm

The End

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

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

Share tips and ideas with other users on

http://forums.perforce.com