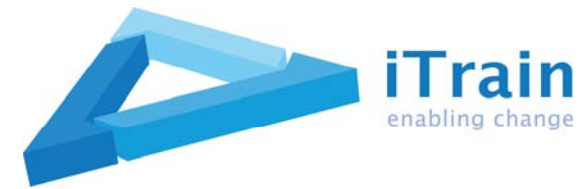


# SSH & Data Pump

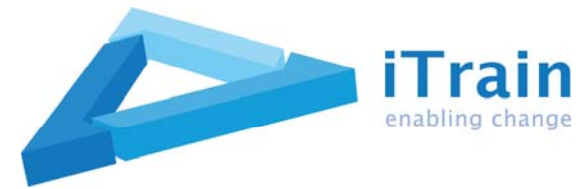
Presented By  
Jeff Cragg  
iTrain

# Some Problems with import and export



- Should an export fail for some reason, you must restart.
- Client based single thread model.
- Dump files must be read completely whether required or not.
- Difficult to measure progress.
- Limited object filtering capabilities.

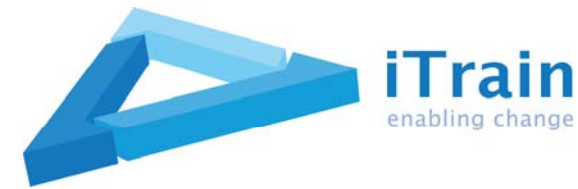
# Data Pump advantages



- Data Pump supplements the existing functions of the import and export tools.
- Introduced with Oracle version 10.1
- Parallel processes to utilise resources.
- Supports Direct Load
- Server based not client based.
- Uses file groups and requires a directory object.
- Self-tuning. Buffer parameter not required.

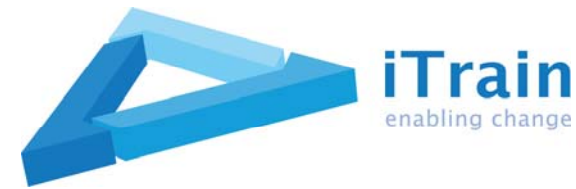
...continued

# Data Pump advantages



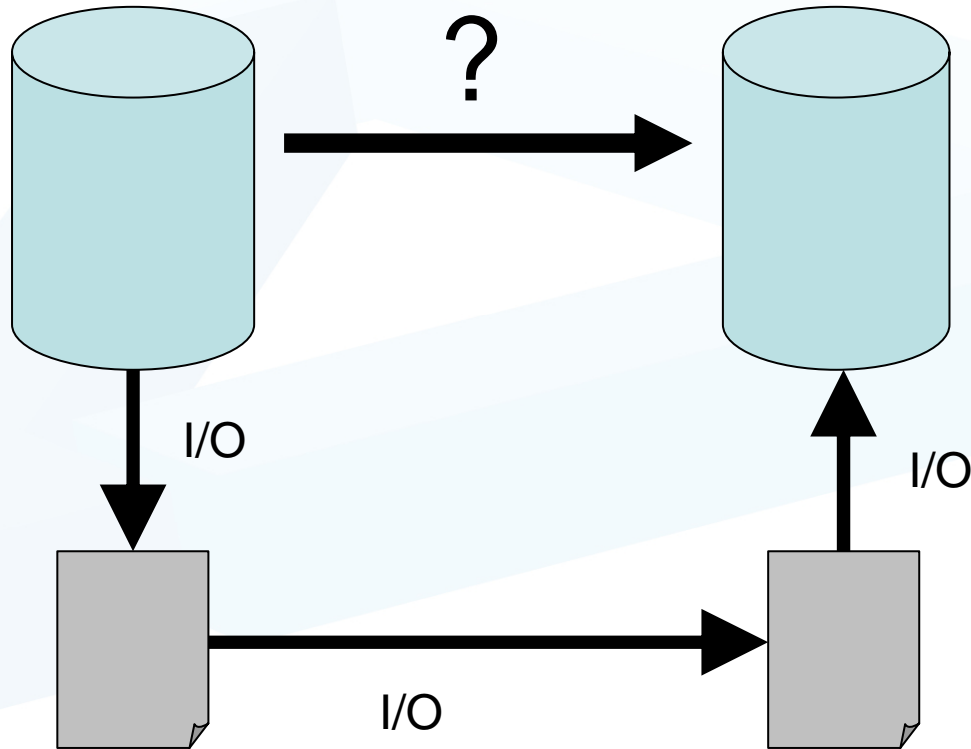
- OS independent.
- You may REMAP schemas, datafiles and tablespaces.
- Callable from PL/SQL.
- Interactive
- May detach and re-attach to jobs.
- Jobs may be started, stopped and resumed.
- Job status available.
- Flashback supported.
- All object types are supported for import and export.

# Some Problems

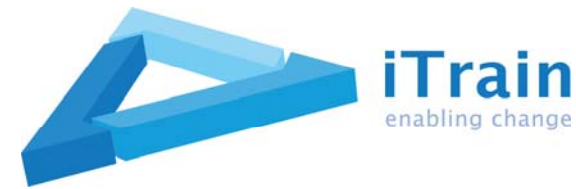


- Compatibility between minor versions only.
- Direct network load not supported between 10.2 and 11G.
- Requires open database link for direct load.
- Original export still required for different versions.
- Database link may not be available.
- Performance claims may not be true in all cases.

# File System Bottleneck.



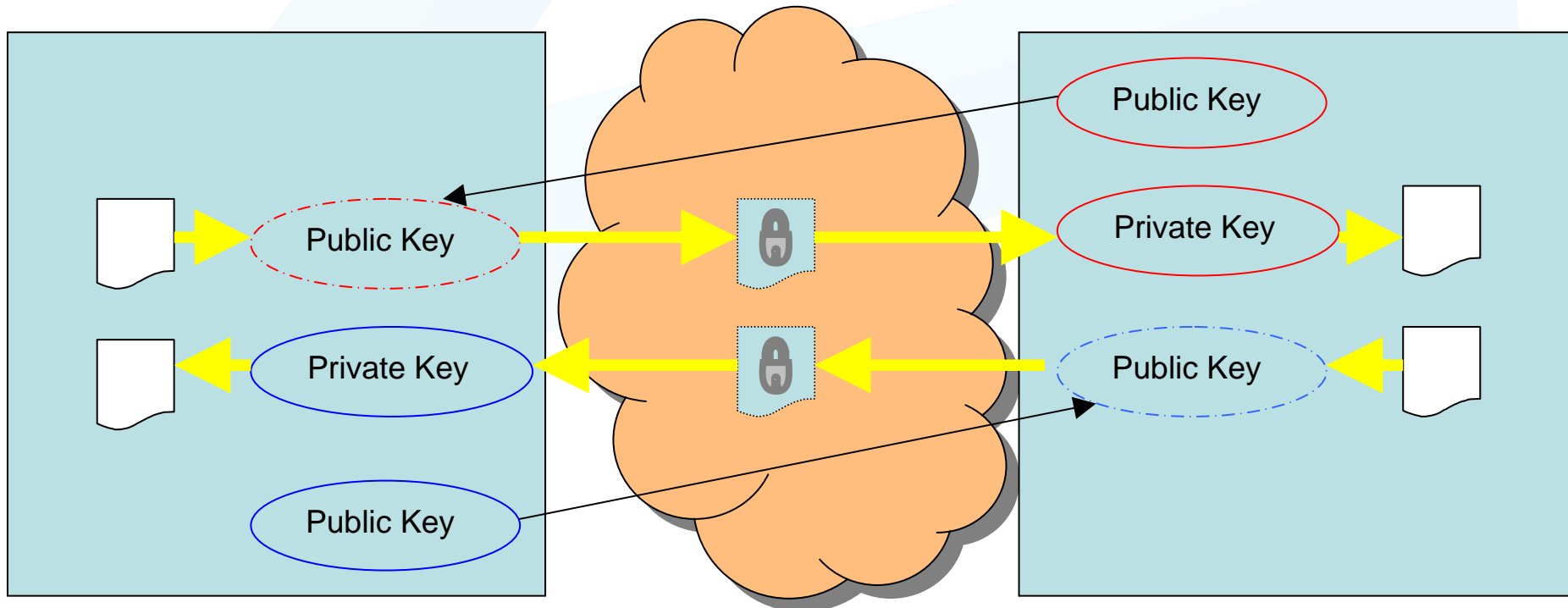
# SSH Fundamentals



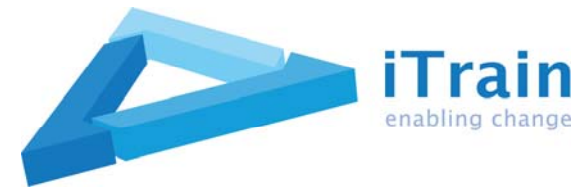
- SSH is a secure enhanced alternative telnet.
- SSH supports tunnelling/redirection.
- SSH supports port forwarding/reversing.
- SSH is supported on all \*nix versions and partially supported on Win32 with the discontinued Cygwin Unix Layer.

# SSH Public-Key Cryptography

- Uses a public and a private key.
- Keys generated automatically.
- The private key decodes messages encoded with the public key.
- The private key may not be derived from the public key.
- Only one public/private key pair is required for a secure channel.

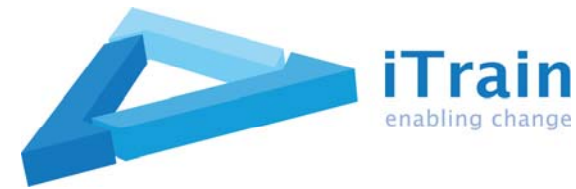


# Sample Problems



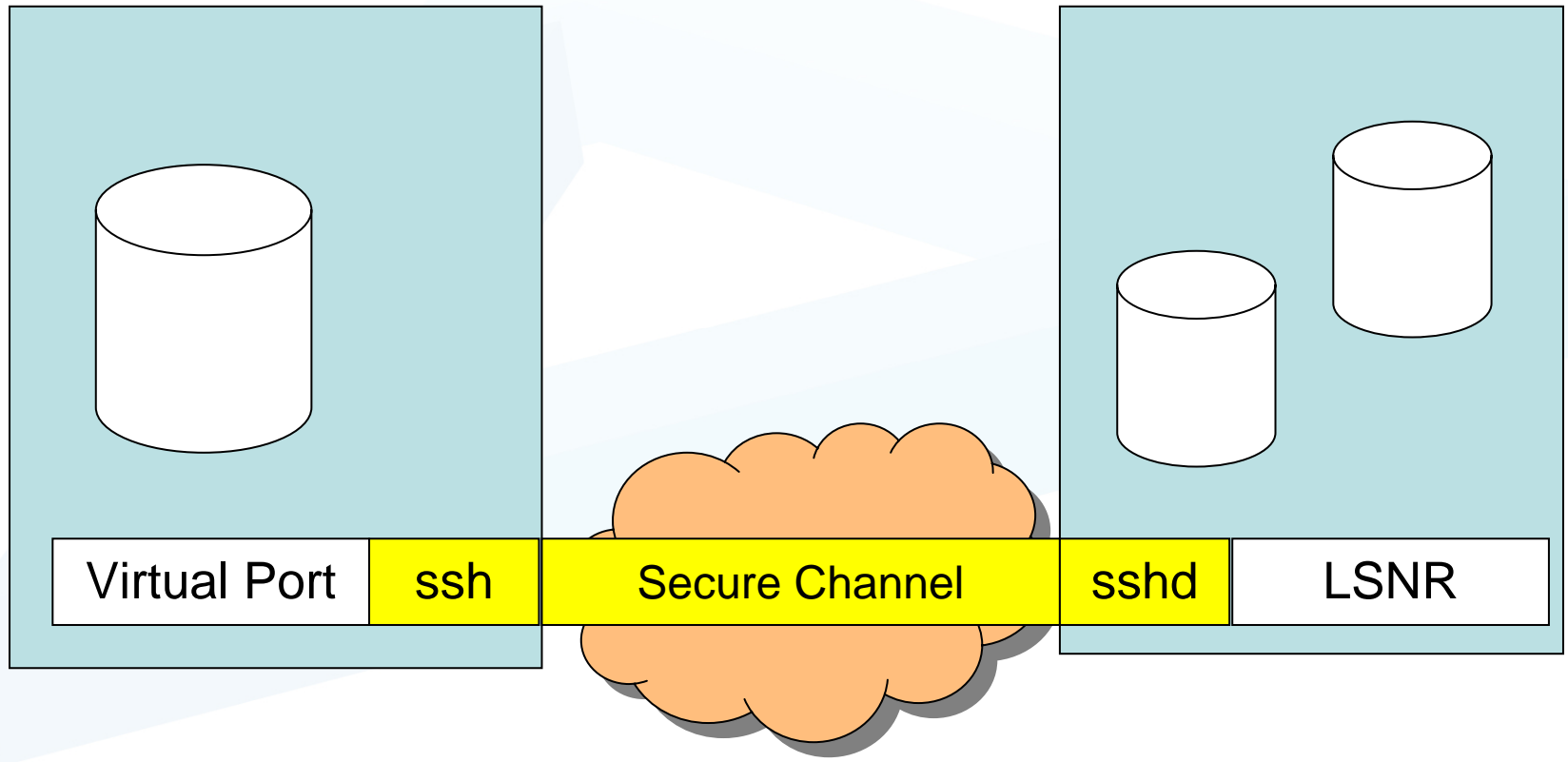
- How do you securely transfer data across an open network with a direct network load without opening the database listener publicly ?
- How do you directly load data from incompatible versions of the database and keep performance ?
- How can you use stream devices with Data Pump ?  
Essentially you can't and probably never will.

## How do you securely transfer data across an open network with a direct network load without opening the database listener publicly ?

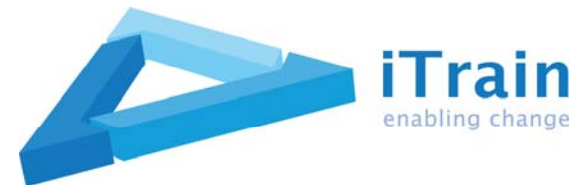


- Start an sshd on the remote database.
- Open a secure connection to the database server.
- Move a port on the local machine to the remote port where the database is listening.
- Create a TNS name for the virtual port.
- Create a database link for the remote database using the virtual port.
- Use Data Pump as usual.
- Data may be directly transferred in an encrypted form.

# Secure Database connection



## Forward the port...

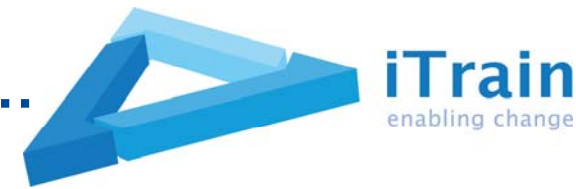


```
>ssh -L 4000:localhost:1521 user@remote.com
```

- Log in with either the Oracle account or use operating system permission structure for greater security.
- Any network traffic directed to localhost:4000 will be magically redirected to port 1521 on the remote machine.
- Open and leave open.
- Both will prompt for a password although a no authentication log in may be setup.

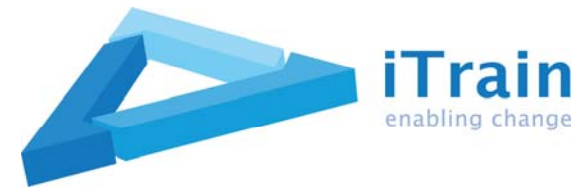
```
>ssh -fN -L 4000:localhost:1521 user@remote.com
```

## Create a TNS entry for the new virtual port...



- Either edit the TNS name file by hand or use the GUI to create an entry with the correct database identifier on the local machine listening on port 4000.
- Ensure that you are modifying the network entries for the server, not just modifying client settings.
- Although expdp/impdp as clients will initiate the job, a specific server will establish an inter-server connection to transfer data, and accordingly must be able to resolve the TNS identifier.
- Check you can access the database through the virtual port with tnsping.

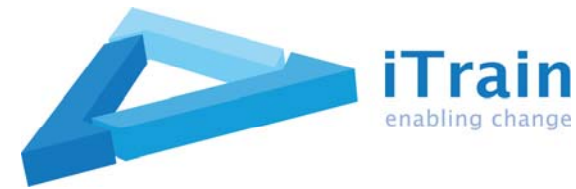
## Create a database link within oracle to the remote database...



```
SQL>create public database link remote connect to user  
identified by password using 'remoteTNS';
```

- Use the same name for the database link as the remote database identifier.
- There are many additional options for database links, including private, user and dual accounts and more.

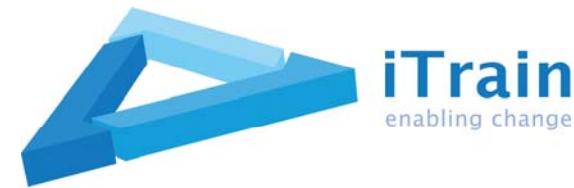
## Create a database directory for dump files and logging.



```
SQL>create or replace directory DATAPUMP as  
' /path/to/directory' ;
```

- You need a directory as operations are server side not client based.
- Even if you are doing a direct load, the directory is needed for logging.
- You may grant read and write access to whomever needs to access this directory in the usual way.

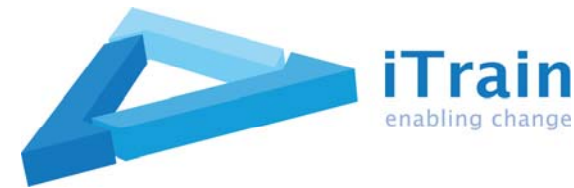
## Use Data Pump as usual: Import



```
SQL>impdp user/password@local tables=remote_table  
directory=datapump network_link=remoteDB;
```

- Utility requires an external directory to be available for logging even though directly transferring from database to database.
- Although network direct export is available, between databases this is always accomplished by importing.
- Exporting from A to B would therefore be performed as an import from B to A.
- Although Data Pump does not support pipes or streams such as tape devices, in this case the details are handled by the database connection.

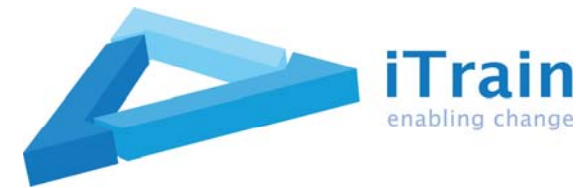
## Use DataPump as usual: Export



```
>expdp user/password@source tables=source_target
directory=datapump dumpfile=exptable.dmp
network_link=remote
```

- This will create a dump file on the remote system in the directory specified.
- The database does the writing, not the client.
- Again, the data passes through the encrypted ssh connection and not to a direct Oracle listener.

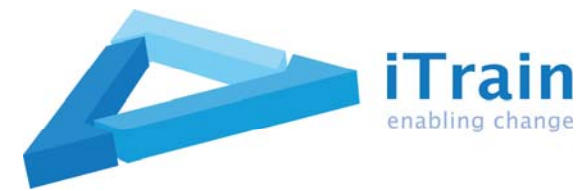
# A Skeletal Data Pump Script



```
ssh -fN -L 4000:localhost:1521 user@remote
expdp user/password@virtualport tables=tablename \
directory=datapump dumpfile=pump.dmp
```

- Assumes the database link and TNS name created.
- Open the secure connection
- Move the port
- Run Data Pump
- **Note that the ssh tunnel created here is persistent and would need to be explicitly closed later if necessary.**
- To run interactively you can press ctrl-c at this point.

# Monitor the Data Pump Job

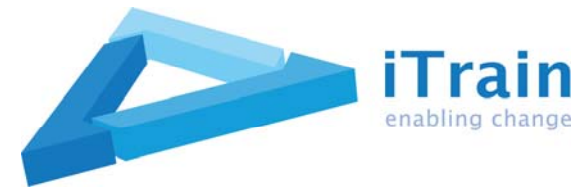


```
SQL> select job_name,job_mode,state from dba_datapump_jobs;
```

JOB_NAME	JOB_MODE	STATE
SYS_IMPORT_TABLE_01	TABLE	EXECUTING

- While the job is in process you can observe the status.
- You may stop a running job by reattaching to the session

## Attach to an existing job.



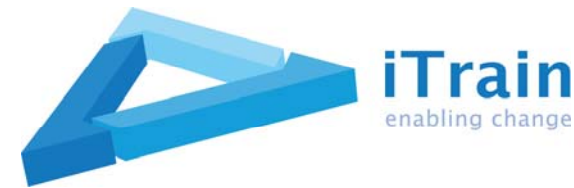
```
>impdp user/password@db attach=sys_import_table_01
```

```
Import> status
```

```
Job: SYS_IMPORT_TABLE_01  
Operation: IMPORT  
Mode: TABLE  
State: IDLING  
Bytes Processed: 0  
Current Parallelism: 1  
Job Error Count:
```

- Attach to the existing job
- Issue commands such as status, stop and start.
- You may increase storage capabilities on a stopped job and restart.

## Closing the ssh tunnel.



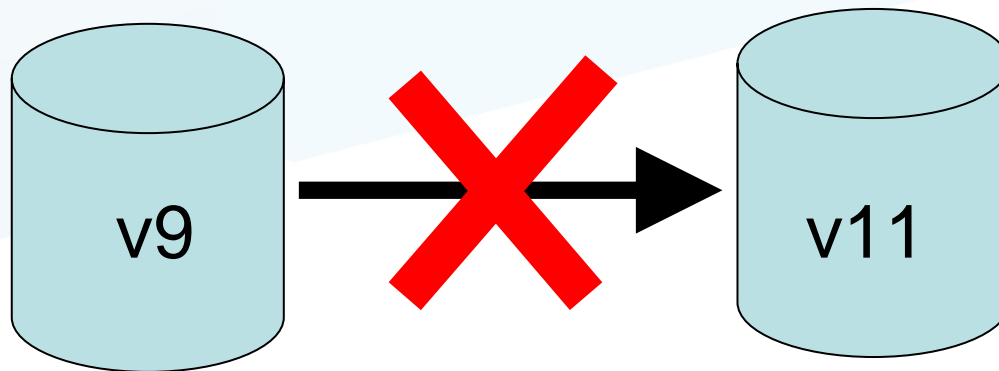
```
>ssh -L 4000:localhost:1521 user@remote ( ls ; ls ) &
```

- If you are running a single command remotely the tunnel will close automatically.
- You can run multiple commands remotely by specifying a subshell.
- If you want to open the tunnel and then run your commands on the local machine you will need to explicitly close the tunnel.
- If you know the time required you can run sleep remotely for a specific time period.

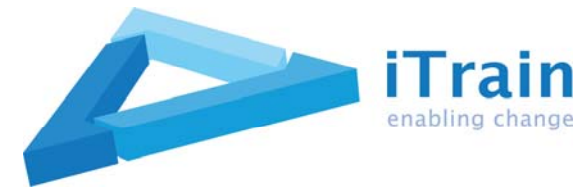
```
>ssh -fN -L 4000:localhost:1521 oracle@remote  
    ...then close with  
>pskill -f "ssh -fN -L 4000:localhost:1521 oracle@remote"
```

# How do you directly load data from incompatible versions of the database and keep performance ?

- Data Pump is not compatible between major versions for the network direct load.
- 10.2 cannot directly load to 11G.
- Versions to 9 are unsupported.
- An alternative method is required.



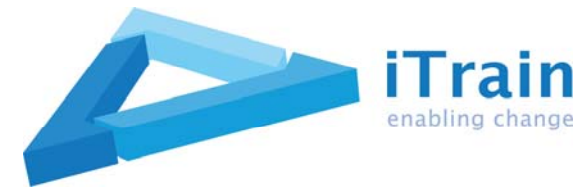
# SSH and Named Pipes. A Fallback Mechanism.



- Pipes can be considered as virtual files available on Unix systems.
- When you write to a pipe the write process will block until a corresponding reader process is started.
- Pipes may be used in conjunction with the redirection facilities of Unix.
- SSH allows redirection across an encrypted network link.

As long as the original export matches the version of the database being exported, a higher version import may then import to a matching higher version database.

# Fallback High Performance Version Tolerant version using export import...

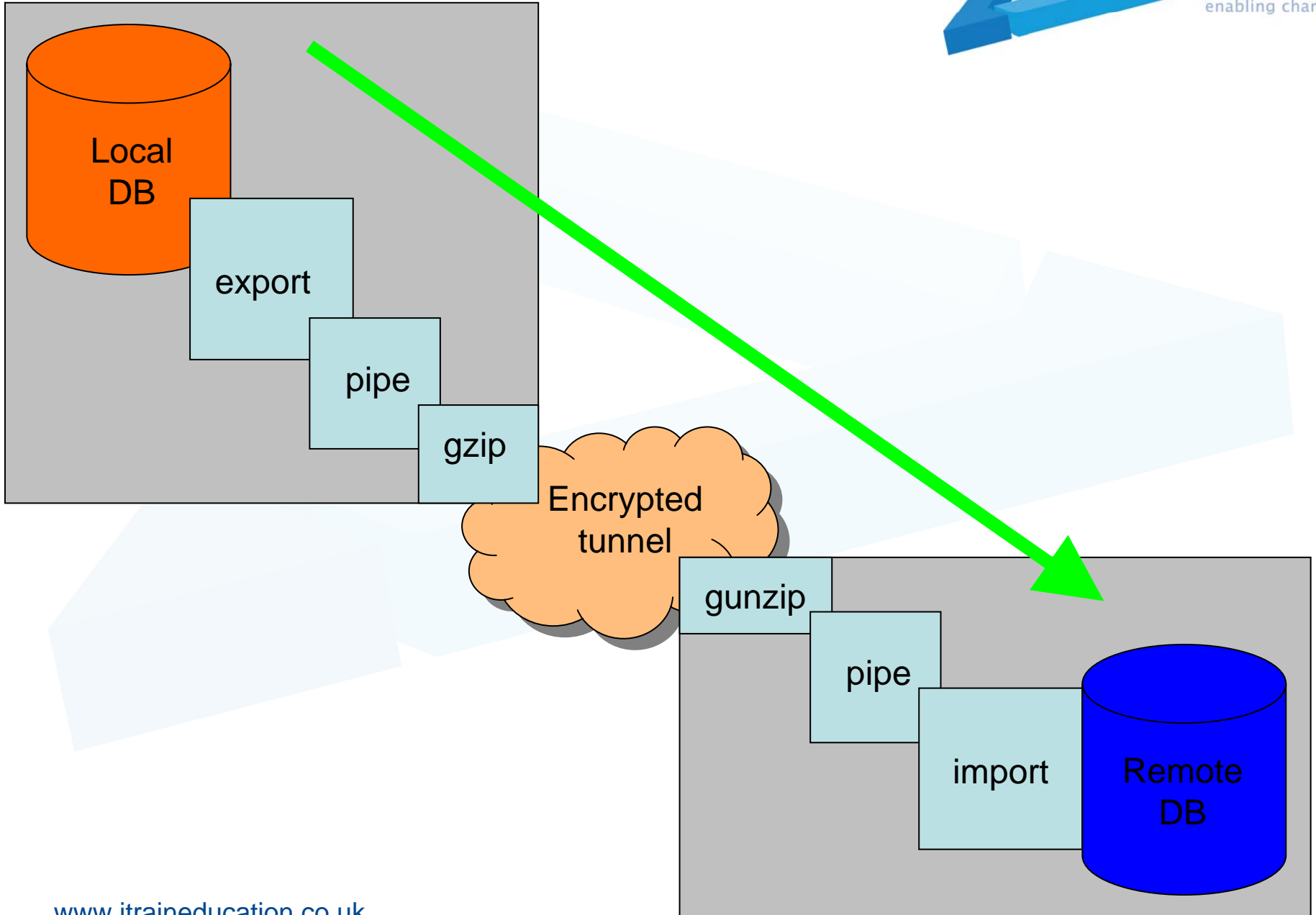


```
echo Starting remote export...
mkfifo ./tempIn
ssh oracle@remoteServer \
'(. ~/.bash_profile ;\
mkfifo ./tempOut ;\
(exp scott/tiger@remote tables=this_is_orcl file=./tempOut &) ;\
gzip -cf <./tempOut ;\
rm ./tempOut )' | gunzip --stdout > ./tempIn &
echo ...starting local import...
imp scott/tiger@orcl2 full=y file=./tempIn
rm ./tempIn
echo ...finished network import.
```

Local Execution

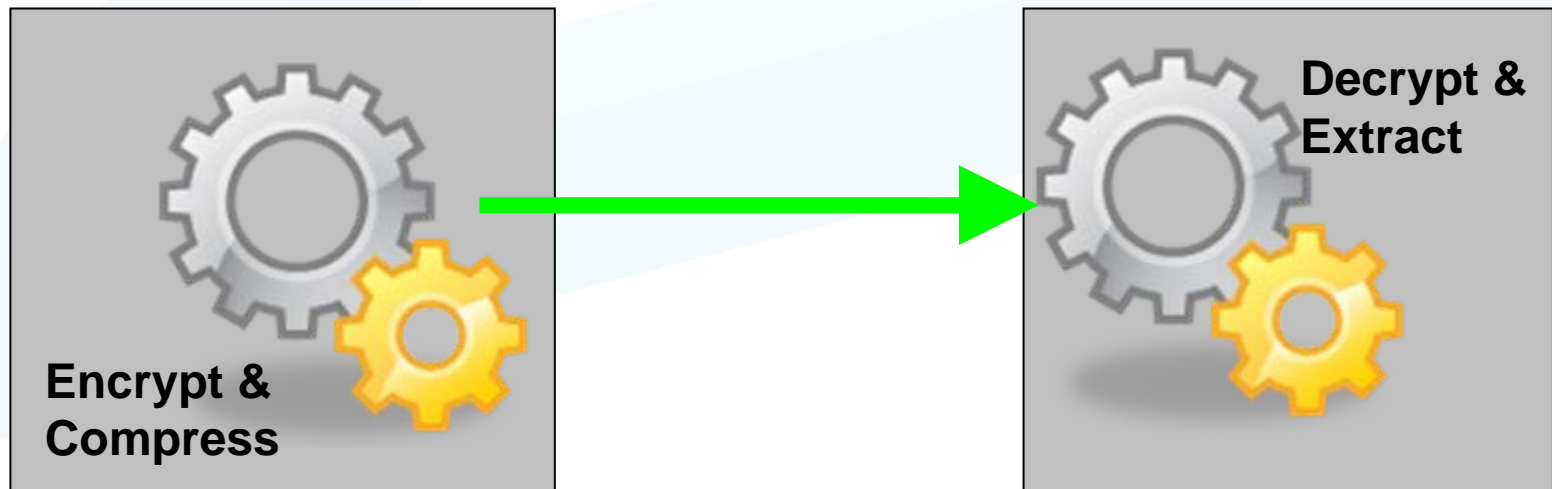
Remote Execution

# Schematic

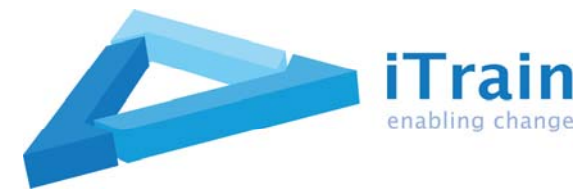


# Why not just use the virtual port?

- Export and import are client based utilities so this is possible.
- The most likely bottleneck in this situation is the network connection.
- A specific compression may be useful for performance.
- In this case, compression/encryption occurs on the remote server and is extracted/decrypted locally, effectively load sharing.

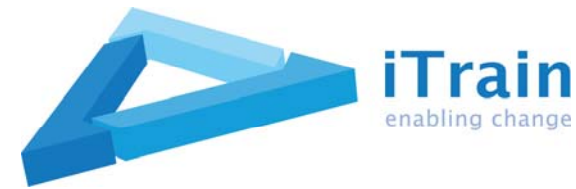


## Additional Options



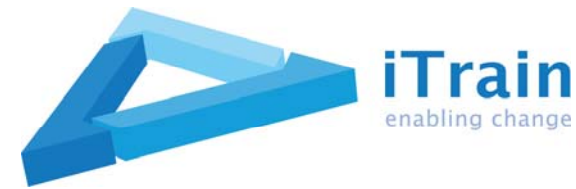
- Zip is used as a placeholder in the example, as ssh will automatically compress traffic using zip with the `-c` option.
- Zip may be replaced with an alternative compression routine such as rar, 7z etc. Additional compression will of course result in less traffic while requiring more CPU time.
- Bespoke encryption may be used in a similar way.
- Although parallel processes such as used by Data Pump are not directly used by export and import, you may start a number of different export/import processes to achieve the same effect if there is no CPU bottleneck.

## Where to go next ?



- Using ssh a network connection may be opened from a fully firewalled network to a remote site. Instead of moving a local port across the network from the local machine, a port on a remote machine may be moved to the local machine.
- The port on that remote site may be used to bypass the firewall, as traffic tunnels back on the outgoing connection and does not need to establish a separate incoming connection.
- The connection need only remain open for as long as required and then closed.

# Conclusion



- Data Pump represents a real advance in functionality over export and import.
- For a site with the same Oracle versions Data Pump offers real advantages.
- If you need to exchange data between different versions of Oracle Data Pump, or you use storage devices which depend on streams, then Oracle Data Pump may not be suitable for you.
- Some features of Data Pump are already available using helpers with export and import.

**Thank you for your time**

**Jeff Cragg  
iTrain  
Stand 32**