ssh server configuration

Standard Blue Light sshd configuration

/etc/ssh/sshd_config

⚠️ Warning: if doing this remotely, when implementing configuration changes, keep the existing ssh session open and test by starting a new one.

⚠️ If a keyword value is set twice, the first value may be effective! Experiment showed that was true for PermitRootLogin and UsePrivilegeSeparation but not for UsePAM. For simplicity we decided not to set any keywords twice (outside Match sections).

There are sshd_config files in the Blue Light git in the conf/ssh directory.

The parameters we change:

- **AcceptEnv** commented out. This ensures no unexpected side effects from having especially non-standard locale variables.
- **AllowUsers** tightens security. We set this to root and add others as the need arises**.**
- **GSSAPIAuthentication** no turns off several authentication methods which are not needed when using private/public keys or passwords. Reference: http://en.wikipedia.org/wiki/Generic_Security_Services_Application_Program_Interface
- **PasswordAuthentication** no or without-password disables login via password.
- **PermitRootLogin** no or without-password disables login via password. We always use without-password, sometimes globally and sometimes within Match Address conditional section(s).
- **UsePrivilegeSeparation** yes. In LXCs only (otherwise we use the as-installed no). Otherwise ssh does not work.
- **UsePAM** no avoids messages like "PAM service(sshd) ignoring max retries; 6 > 3".
  ℹ️ The message is caused by PAM's compiled-in retry limit being less than sshd's.
- **UseDNS** no disables reverse DNS lookups to see if your hostname matches the IP address you are connecting from. Does not make sense with dynamic IP addresses.

/etc/ssh/sshd_config generation

ssh package installation generates /etc/ssh/sshd_config (a change from the original package installation which copied the file) so the as-installed file may not be identical on all systems.

Editing policy

As far as practicable, for easy identification, Blue light changes are made outside the package-generated file's text block and the original text block is surrounded by:

```
# ==== THE ORIGINAL /etc/ssh/sshd_config, edited, STARTS HERE ====
...
# ==== THE ORIGINAL /etc/ssh/sshd_config, edited, ENDS HERE ====
```

Within that original block, the only changes should be to comment out keywords. To aid identification they can be commented out with a #@ prefix.

After the original block ...

```
# Blue Light changes to as-installed settings (should be commented out above)
PermitRootLogin without-password     # Comment out if using the Match Address stanzas below
UsePAM no
UsePrivilegeSeparation no            # Required in an lxc container

# Blue Light additions to change as-installed settings not specified above (defaults)
AllowUsers root
PasswordAuthentication no

# Blue Light additions to improve performance
UseDNS no
GSSAPIAuthentication no
```

Match Address stanzas

We are still evolving how we use these. TODO: does the order matter (does sshd stop processing these stanzas after the first match?)? TODO: do these PermitRootLogin values override any global setting?

Some samples:

```
# Allow root login from the BLUE OpenVPN addresses
Match Address 10.42.23.0/24
    PermitRootLogin without-password

# Allow root login from the BL OpenVPN server
Match Address 10.42.0.1
    PermitRootLogin without-password

# Allow root login from blav2
Match Address 148.251.233.235
    PermitRootLogin without-password
Match Address 10.42.0.2
    PermitRootLogin without-password
```

/etc/default/ssh

```
root@localhost:~# diff /etc/default/ssh{.org,}
5c5
< SSHD_OPTS=
---
> SSHD_OPTS=-u0
```

Warning: if doing this remotely, keep the existing ssh session open and test by starting a new one.

Enable the new configuration by `service ssh restart`

authorized_keys

command=

Security can be tightened by specifying the command that an authorised key can run by inserting command=<command> before the key itself. Details in the sshd man page, in the command="command" section.

Sometimes this is not convenient because several commands are to be run. A solution is to specify a script in authorized_keys and for the script to validate the commands. For example (old version of bung's validate_ssh_cmd.sh):

```
#!/bin/bash
# Purpose: validates the command a remote host is attempting to execute by ss

df_regex='^df '
rsync_server_regex='^rsync --server '
stat_regex='^stat --format=%F '

if [[ $SSH_ORIGINAL_COMMAND =~ $df_regex \
    || $SSH_ORIGINAL_COMMAND =~ $rsync_server_regex \
    || $SSH_ORIGINAL_COMMAND =~ $stat_regex \
]]; then
    exec $SSH_ORIGINAL_COMMAND
else
    echo "${0##*/}: command did not pass validation ($SSH_ORIGINAL_COMMAND)" >&2
    exit 1
fi
```

TODO: enhance the script to read the regexes from a config file.