Description of problem

There is a cluster of servers. We have a new developer joining the team who needs access to these and we need to revoke access if needed. But in an automated way!

Description of solution

For this problem I have developed two scripts.

One script for granting access to the developer, the other script is for revoking their access.

Design and Reasoning.

The first script for granting access uses the “auto\_ssh.sh” script, this generates a SSH key on the local host and copies the key to the remote host.

First this assumes that the developer user account has been created on the remote host already, this script matches the key generated on the local host with the developer account on the remote host. Thus allowing the developer to login to the remote host using an SSH command, as opposed to creating and managing passwords.

The choice of using SSH key generation is due to its tighter security capabilities as opposed to passwords which can be weak or duplicated by users. SSH keys are randomly generated strings which are matched on the local and remote hosts to allow access to them without password authentication.

Further reasoning for using SSH key authentication is that generated keys can have enhanced security with longer bit counts making them harder to decrypt (see future road map for details). SSH keys authentication can be scaled across multiple servers. As long as the local host and remote host keys match on the various servers, users can access the remote host by SSH’ing, reducing the need for multiple passwords (See future road map for details on this).

The second script simply SSH’s to the remote host and deletes the desired user from the remote host. This removes the user and their paired keys on the remote host so when they try to SSH to the desired remote host, it will deny them access by asking for a password which does not exist.

Tech Used

* Azure - Running an Ubuntu VM, this is the remote host.
* Local machine – Windows and Mac machine for accessing the VM on both public and corporate network.
* Git bash - CLI window used to run the automated script.
* Brackets - Developer suite for creating the script.

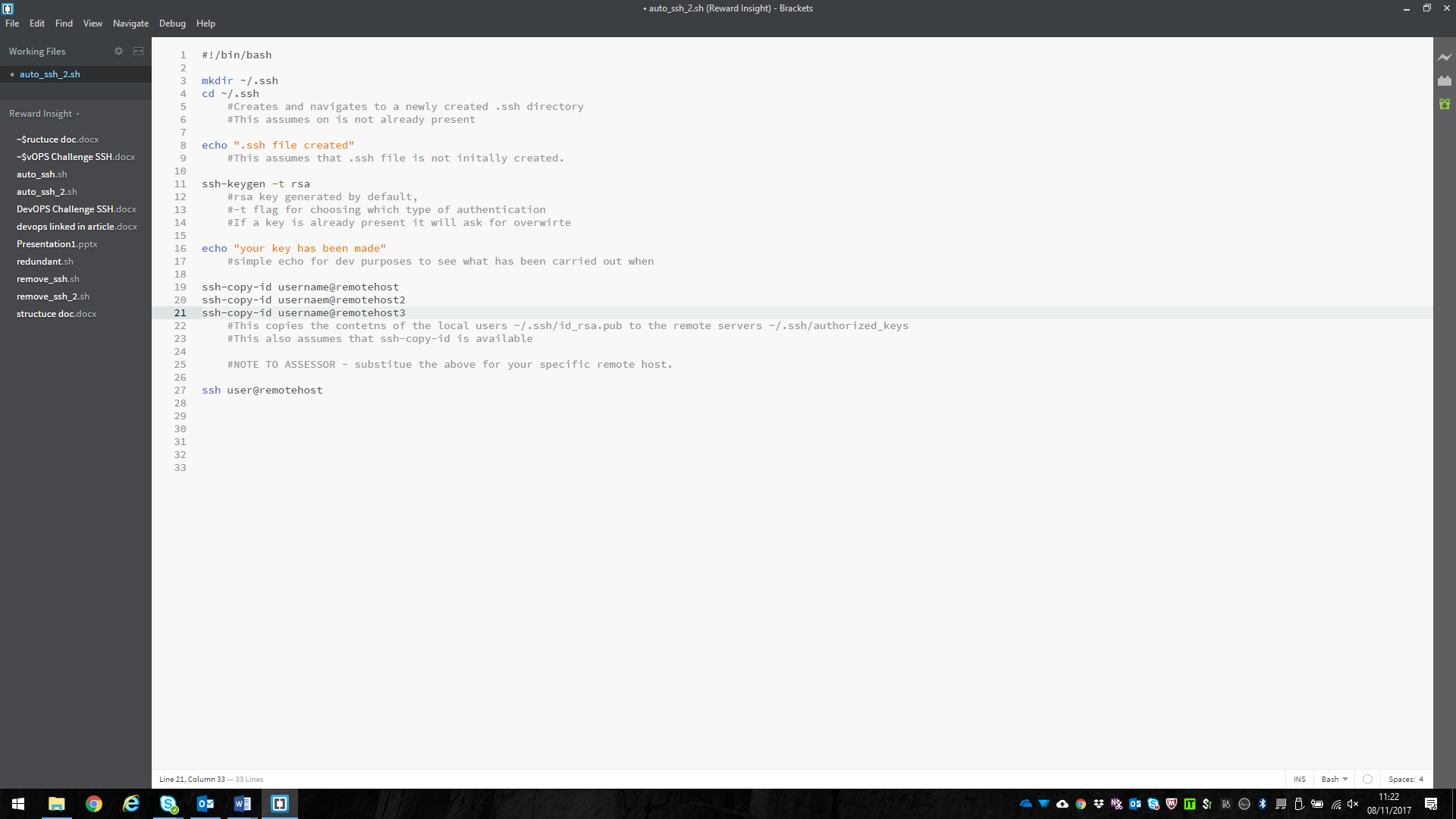
Assumptions

* The Developer account is already set up on the remote host.
* There are no proxy’s, firewalls or security boundaries.
* There are no jump boxes in place between local developer machine and the remote host.

*In script assumptions*

* The script assumes ~/.ssh is not already created. However if so, only a minor warning appears.
* The script assumes you want RSA authentication, this can be edited for preference.
* The script assumes you have ssh-copy-id available.

Future road map

* With the current assumption that the developer account is already set up on the remote host. Going forward the script could incorporate the creation of a new user with all the suitable access rights and permissions attached in it for further automation.
* As mentioned in the design section, SSH Key generation can have increased security capabilities by lengthening the bit count of the generated key.
  + This is done by the –b flag at the keygen command, i.e. ssh-keygen –b 4096
* As mentioned in the design section the scalability of user access across a cluster of servers is possible. The generated key on the local host has to match the one on the desired remote host. Where this script only does it for one remote host, one would have to edit the script to include the other servers (see screenshot for example).
* As the expansion of a system or a team happens, a CMS (configuration management system) could be used, such as Ansible, Puppet, Chef. These tools can be employed to keep uniform and consistent configuration cross a system.
  + For this example an Ansible play book or chef cookbook could be created with all the user accounts that need creating along with associated access and permission rights, and deployed out to a number of remote hosts in a quick, efficient and uniformed style.

**Developer Walkthrough**

See SSH’ing Walkthrough document.