view-deployment

Description

A Collection of Ansible deployement scripts for the *View Boston* installation.

The scripts can be run from any machine (Linux, macOS or Windows under WSL) connected to the network. They will configure and deployed the needed resources on the targets.

Setup on target machines

A setup script is included to setup the WinRM remoting service and take care of the authentication protocols.

Enable running PowerShell Scripts

In Powershell as Administrator:

```
SetExecution-Policy Unrestricted
```

Download and run setup script in Powershell

```
[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12
$url = "https://raw.githubusercontent.com/ProxeeSolutions/view-
deployment/main/view-win-host-setup.ps1"
$file = "$env:temp\view-win-host-setup.ps1"
(New-Object -TypeName System.Net.WebClient).DownloadFile($url, $file)
powershell.exe -ExecutionPolicy ByPass -File $file
```

Setup

Windows Subsystem for Linux

Ansible only runs under WSL in Windows. From an elevated PowerShell prompt:

```
wsl --install
```

Once completed, restart the computer. Download the latest Ubuntu LTS release from the Microsoft Store. It will then be available as `ubuntu` from the Windows menu.

The current installation is running from the `SVR01-03` machine. the scripts are in the `~\view-deployement` directory.

Install Ansible

```
sudo apt install python3
pip install ansible
```

Test in Ansible

On the Ansible client host:

```
ansible all -i 192.168.236.218, --user vagrant --ask-pass -m win_ping -c
winrm -e "ansible_winrm_server_cert_validation=ignore"
```

It should return:

```
192.168.236.218 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Usage

Target groups in inventory

the `inventory` file contains the definition for the hosts and some hosts variables.

- lobby
- poster
- interactive
- cms
- storage
- finale
- elevator
- viewer
- tactile

Group variables

Insidee the `group_vars` directory, group variables can be specified for specific groups by putting then in the corresponding `<group>.yml` file

Secrets

Secret passwords and token are store in an encrypted vault under the `groups_vars/all` directory

File share

To deploy the necessary files on the stations, ansible will use a windows file share located on the storage server at `\\192.168.10.26\ViewBoston`. This share will be synced to every machine during the deployment process.

To update the Electron app, place the new version in the following path `\\192.168.10.26\ViewBoston\Electron\`and append the date (and a letter for multiple version in the day) to the executable file. For example `C:\ViewBoston\Electron\20230323b_ViewBostonSetup1.0.0.exe` would correspond the the second version for the 23 march 2023. You must then change the reference in the following deployment script: `roles/electron/tasks/install-electron.yml` on line 10.

Running Ad-Hoc command

To issue on-demand command, use the `ansible` command. the `-m win_shell` option will be used to send shell commands to windows machines. The target can be one or multiople machines separated by comas and a trailing coma, or a group defined in the `inventory` file.

```
ansible -m win_shell -a '<windows shell command>' <target>
```

Ad-Hoc Examples

Ad-hoc commands can be run without the need for a playbook file.

PM2 restart on poster machines

```
ansible -m win_shell -a 'pm2 restart all' poster
```

Reboot poster Computers

```
ansible -m ansible.windows.win_reboot poster
```

Running playbooks

Running playbooks is the automated way of executing every actions on the machines. It can be run many times and will only process the required tasks to bring the machines to the desired state.

Actions can be filtered by hosts or groups using the `--limit` option like the adphoc command's target option. If unspecified, all hosts in the inventory will be targeted, including the servers.

Actions for the whole site can be found in the `site.yml` playbook, which imports specific sub-playbooks for every group. Tags can also be used to only execute tasks which have been tagged with a particular identifier.

Variables for groups and hosts are specified in the group files (*.yml) under the `group_vars` directory

The command is:

```
ansible-playbook <playbook.yml> (optional: --tags: <list of comma-
separated tags>) (optional: --limit <comma-separated groups or hosts>)
```

Playbook Examples

Run the playbook for the whole site, setting the proper configuration on every machine:

```
ansible-playbook site.yml
```

Run the playbook only on a specific group:

```
ansible-playbook site.yml --limit tactile
```

Only synchronize files in the `ViewBoston/` directory from the storage server:

```
ansible-playbook site.yml --tags sync
```

Set audio volumes everywhere (volume variable taken from the `group_vars` files)

```
ansible-playbook site.yml --tags audio
```

Set audio volumes for sound effects and tts using tags

```
ansible-playbook site.yml --tags: electron,audio
```

If the playbook hangs on the Sync View Boston directory task, it means the SMB server on SRV-03 has maxed out it's connection. Reboot it to resolve.

Purging Request Cacher public files

```
ansible-playbook site.yml --limit <groups or machines> --extra-vars
'{"purge_requests": true}'
```

Purging ViewBoston files limiting to interactives

```
ansible-playbook site.yml --limit interactive --extra-vars
'{"purge_viewboston": true}'
```

Rebooting after site deployment

```
ansible-playbook site.yml --extra-vars '{"reboot_machines": true}'
```