Managing Waldur with Ansible

NB! Repository with Ansible playbooks for Waldur management is not open-sourced. It is available to Waldur users that have purchased support packages.

Compatibility

Ansible version 2.9 is supported; code in this repository may work with other Ansible versions but it is not guaranteed.

Quick setup

1. Make sure that you have:
    - Folder containing managed-ansible Ansible installer (this one).
    - Folder containing deployment-specific settings.
2. Copy `setup_deployment.sh.example` to `setup_deployment.sh`, adjust:
    - INSTALLER_PATH - full path to current folder
    - DEPLOYMENT_CONFIG_PATH - full path to folder with deployment-specific information
    - DEPLOYMENT_IDS - list of deployment IDs you want to manage.
3. Run ./setup_deployment.sh. This will create required symlinks.
4. Run `ansible-playbook -DC <deployment_id.yml>` to check what installer will do.
5. Run `ansible-playbook -D <deployment_id.yml>` to apply the installer.

Upgrading installer

1. Download a new archive into a separate folder and unpack.
2. Check what has changed: `rsync --dry-run -avzh unpacked-folder-eg-ansible-3.2.3/ /path/to/installer/`
3. Do the upgrade: `rsync -avzh unpacked-folder-eg-ansible-3.2.3/ /path/to/installer/`

Upgrading Waldur

Upgrading Waldur to a new version is achieved by following the checklist:

1. Update deployment-specific variables for the new version in `groups_vars/<deployment_id>/vars` and `groups_vars/<deployment_id>/vault` (if setting is private).
2. Update target version in `groups_vars/<deployment_id>/vars`: set `waldur_homeport_version` and `waldur_mastermind_version` to a new version of the Waldur release.
3. Run `ansible-playbook -DC <deployment_id.yml>` to check what installer will do during the upgrade.
4. Run `ansible-playbook -D <deployment_id.yml>` to perform an upgrade.

Add new deployment

Infrastructure:

1. Prepare servers matching requirements.
2. Make sure you can access them by SSH by running only `ssh <host>`. Otherwise you should tweak `~/.ssh/config` file or describe connectivity in the Ansible inventory.

Ansible (this repository):

1. Add new deployment-specific host group to `hosts` file (example: `[foo]`); add hosts to this group.
2. Add hosts to Waldur role groups in `hosts` file.
3. Add variable files for new deployment: `group_vars/foo/vars` and `group_vars/foo/vault`
4. Add deployment specific information under `deployments` folder.
5. Copy existing playbook that is the closest match for a new deployment; modify as needed
6. Run Ansible playbook to set up deployment: `ansible-playbook -D foo.yml`

Remove deployment

Ansible (this repository):

1. Delete deployment-specific playbook, roles, tasks, templates etc.
2. Delete variable files: `rm -rf group_vars/foo`
3. Delete deployment-specific host group and all deployment-specific hosts from `hosts` file (example: `[foo]`)

Managing Waldur deployed as Helm

Requirements:

1. Installed and running `kubernetes` system on a node (e. g. minikube)
2. Installed `kubectl` on the node

Place all configuration files for release in next manner:

- `values.yaml` -> `roles/waldur_helm/files/waldur_helm/waldur/`
- `values.yaml` -> `roles/waldur_helm/files/waldur_helm/waldur/`
- Files related to TLS -> `deployments/<deployment_id>/tls/`
- Files related to white-labeling -> `deployments/<deployment_id>/whitelabeling/`
- Files related to mastermind templates -> `deployments/<deployment_id>/mastermind_templates/`
- Files related to stress testing -> `deployments/<deployment_id>/locust_tasks/`
- Files related to SAML2 -> `deployments/<deployment_id>/waldur_saml2/`

More configuration info:

- TLS
- White-labeling
- Mastermind templates

Last update: 2021-04-25