TON OS DApp Server configuration environment

Repository contains automated configuration based on TON OS DApp Server in TON blockchain repository.

System Requirements

Configuration	CPU (cores)	RAM (GiB)	Storage (GiB)	Network (Gbit/s)
Recommended	24	192	2000	1

SSD disks are recommended for the storage.

Current restrictions

Infrastructure:

- DigitalOcean cloud
- Physical server
- Single-server infrastructure and deployment

Deployment:

- Ubuntu LTS Server 16.04/18.04/20.04 at server side
- Legacy docker-compose deployment

Operator:

- GNU/Linux at operator side

Prerequisites

Consider latest versions:

- `openssh-client` (for server configuration deployment)
- `ssh-agent` (ssh keys in-memory storage)
- `terraform` (infrastructure provisioning)
- `Ansible` (software provisioning and deployment)
- `git` (configuration synchronization)

Usage

Fork

This may look misleading but you need to keep your customized configuration somewhere under version control, e.g. in private repository.

Update environment

The environment files are:

- `./ansible/group_vars/tonos`. Here you should set your configuration single source of truth at `config_repo`
- `./scripts/env.sh`

Clone

Note: you will probably need to clone your fork and not the original repository if you require any configuration update.

```
$ git clone https://github.com/amttr/tonos-config.git ~/tonos-config
```

SSH keys

You will need two keys: one for server access and other for git server access at server side. Or you can use single one for both for simplification.
In a case of a dedicated server it is possible that you're already have one.

1. Generate keys

   ```
   $ ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa_tonos
   ```

2. Initialize `ssh-agent`. If you haven't used it yet, consider to add it to your terminal emulator configuration

```
$ eval "$(ssh-agent -s)"
```

3. Add your keys to the `ssh-agent` with `ssh-add`

```
$ ssh-add ~/.ssh/id_rsa_tonos
```

Infrastructure instantiation

Dedicated server

Suppose you already have some server with required OS and resources. The only thing left before provisioning/deployment is to generate `inventory` description file for Ansible.

```
$ cd ~/tonos-config/infra/dedicated
$ ./build_inventory.sh <server ip> \
                       <server user>  \
                       <git user ssh id path>
```

DigitalOcean

1. Create an API token
   DO: Personal access token howto)
2. Choose your region, instance type and base image
   To define your needs, you will need a DigitalOcean control tool `doctl`.
   Regions:

```
$ doctl compute region list

Slug    Name               Available
...
fra1    Frankfurt 1        true
...
```

   Instances:

```
$ doctl compute size list

Slug               Memory    VCPUs    Disk    Price Monthly    Price Hourly
...
s-4vcpu-8gb        8192      4        160     40.00            0.059520
...
```

Base images:

```
$ doctl compute image list --public | grep ubuntu

ID          Name                                                    Type        Distribut

...
68629515    20.04 (LTS) x64                                         snapshot    Ubuntu
...
```

3. Create personal terraform variables file (`~/tonos-config/infra/digitalocean/terraform.tfvars`)

Key	Description
`do_token`	DigitalOcean personal access token
`user_ssh_id_path`	Remote user ssh id path
`git_ssh_id_path`	Remote git user ssh id path
`image`	Chosen image with supported OS
`region`	Server location region
`size`	Instance type

Example:

```
do_token         = "youshouldkeepthatsecret"
user_ssh_id_path = "~/.ssh/id_rsa_tonos"
git_ssh_id_path  = "~/.ssh/id_rsa_tonos"
image            = "ubuntu-20-04-x64"
region           = "fra1"
size             = "s-4vcpu-8gb"
```

3. Install `terraform` provider

Terraform use providers for interractions with different clouds. In this case it's DO provider. To install it:

```
$ cd ~/tonos-config/infra/digitalocean
$ terraform init
```

4. Set-up an infrasture
   Simply:

   ```
   $ cd ~/tonos-config/infra/digitalocean
   $ terraform apply
   ```

   Or in case you will need to tear everything down, for instance type upgrade for example:

   ```
   $ cd ~/tonos-config/infra/digitalocean
   $ terraform destroy
   ```

Configuration

Dedicated

To provision, just execute a corresponding playbook:

```
$ cd ~/tonos-config/ansible
$ ansible-playbook -vv playbooks/provision.yml
```

Cloud

In case of a cloud inital software instantiation is already done by `terraform`.

Deployment

```
$ cd ~/tonos-config/ansible
$ ansible-playbook -vv playbooks/deploy.yml
```

Monitoring

TBD

Improvements TODOs

- Monitoring
- Separate user support
- VPC and single point perimeter access (lb+bastion)
- Firewalling
- Different cloud platforms support
- Monitoring and DApp Server separation