

ITI8801 · Cloud Architectures and DevOps

Pointers for starting the teamwork project

Karl-Andreas Turvas, TalTech, Autumn 2026

Agenda

1. ETAIS intro (the cloud you will be using)
2. Waldur and its clients (the backend ETAIS uses)
3. Some tips

ETAIS: Estonian Scientific Computing Infrastructure

Homepage for info: etais.ee

Web dashboard for compute resources: minu.etais.ee

What is ETAIS?

Pan-Estonian **unified** e-infrastructure that offers users **cloud services**, classical scientific computing (HPC - high-performance computing), hosting and management services for scientific web applications, and resources for repositories

ETAIS: Your teamwork resources

You will be interacting with a small section of ETAIS: the TalTech **cloud services** part.

You will be added to a 'Project' with a 'Private Cloud' in which you can create VMs, subnets, etc.

Note: It's a lot more simplistic than Azure/AWS/the other big dogs

ETAIS: Demo

Let me show you around a bit

Waldur

Backend for ETAIS:

- Waldur GitHub: github.com/waldur
- Docs: docs.waldur.com

- API endpoint: minu.etais.ee/api
- API docs: api-docs.waldur.com

Waldur: Clients

Waldur has many clients:

- Python client: github.com/waldur/py-client
- Go client: github.com/waldur/go-client
- Rust client: github.com/waldur/rs-client
- JavaScript client: github.com/waldur/js-client
- Terraform provider: github.com/waldur/terraform-provider-waldur
  - Siim Rebane's patched Terraform provider: github.com/siimrebane/terraform-provider-waldur
- Ansible module: github.com/waldur/ansible-waldur-module-next

So which ones to use?

Up to you to figure out which one works the best for you!

Couple of examples on next slides

ETAIS provisioning: Terraform task example

registry.terraform.io/providers/waldur/waldur/latest/docs/resources/openstack_instance

```
resource "waldur_openstack_instance" "example" {
  count = 40 # Creates 40 VMs, like the Ansible loop

  name                = format("iti0302-2025-%02d", count.index + 1)
  flavor              = data.waldur_openstack_flavor.example.url
  image               = data.waldur_openstack_image.example.url
  project             = data.waldur_structure_project.example.url
  offering            = data.waldur_marketplace_offering.example.url
  ssh_public_key      = data.waldur_core_ssh_public_key.example.url
  system_volume_size = data.system_volume_size
  ports = [
    {
      subnet = data.waldur_openstack_subnet.example.url
    }
  ]
}
```

ETAIS provisioning: Ansible task example

gitlab.hpc.taltech.ee/hpc-public/ansible/ansible-etais-provisioning

(only accessible from TalTech networks)

```yaml
- name: Create VM instances
  waldur_marketplace_os_instance:
    api_url: "{{ api_url }}"
    access_token: "{{ access_token }}" # Set in ENV variables
    project: "{{ project }}"
    offering: "{{ offering }}"
    name: "iti0302-2025-{{ '%02d' | format(item) }}" # VM names: iti0302-2025-01, iti0302-2025-02, ...
    flavor: "{{ flavor }}"
    image: "{{ image }}"
    subnet: "{{ subnet }}"
    ssh_key: "{{ ssh_key }}"
    system_volume_size: "{{ system_volume_size }}"
    system_volume_type: "rbd"
    floating_ip: auto
    wait: true # Waits for the instance to be active before proceeding to the next VM creation
    state: present # Set to 'present' to create, set to 'absent' to delete
    security_groups: "{{ security_groups }}"
  loop: "{{ range(1, 41) | list }}" # Creates 40 VMs in total
  loop_control:
    label: "Creating VM iti0302-2025-{{ '%02d' | format(item) }}" # Log output
```

Terraform: GitLab CI example

```
.terraform:build:
  image:
    name: $TF_IMAGE
  stage: build
  script:
    - echo "$TF_PLAN_ARGS"
    - gitlab-terraform plan $TF_PLAN_ARGS
    - gitlab-terraform plan-json $TF_PLAN_ARGS
  resource_group: ${TF_STATE_NAME}
  artifacts:
    paths:
      - ${TF_ROOT}/plan.cache
    reports:
      terraform: ${TF_ROOT}/plan.json

.terraform:deploy:
  image:
    name: $TF_IMAGE
  stage: deploy
  script:
    - gitlab-terraform apply
  resource_group: ${TF_STATE_NAME}
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH && $TF_AUTO_DEPLOY == "true"
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
      when: manual
```

General tips

- **Everything must be IaC!** But layers use be different repos and tech
  - Example:
    1. Terraform repo for provisioning
    2. Ansible repo for OS/service configure
    3. Application repo with code and Dockerfile
- **You shouldn't need more than 2 public IPs**: 1 for admin VPN and 1 for reverse proxy/load-balancer
- **Figure out load-balancing, reverse proxy** yourself, this cloud does not provide ready made load-balancing
- **Restrict your admin access** to TalTech networks or your home IP addresses

Thank you!

Any questions?

Contact me over email or Teams: **katurv@taltech.ee**

And come to practicals to talk/ask!