# Git:

Centralized vs disturbed git repos.

* Centralized: code with no history of metadata locally. Example SVN.
* Distributed: Code with history and metadata locally. Example Git.

Both remote &local

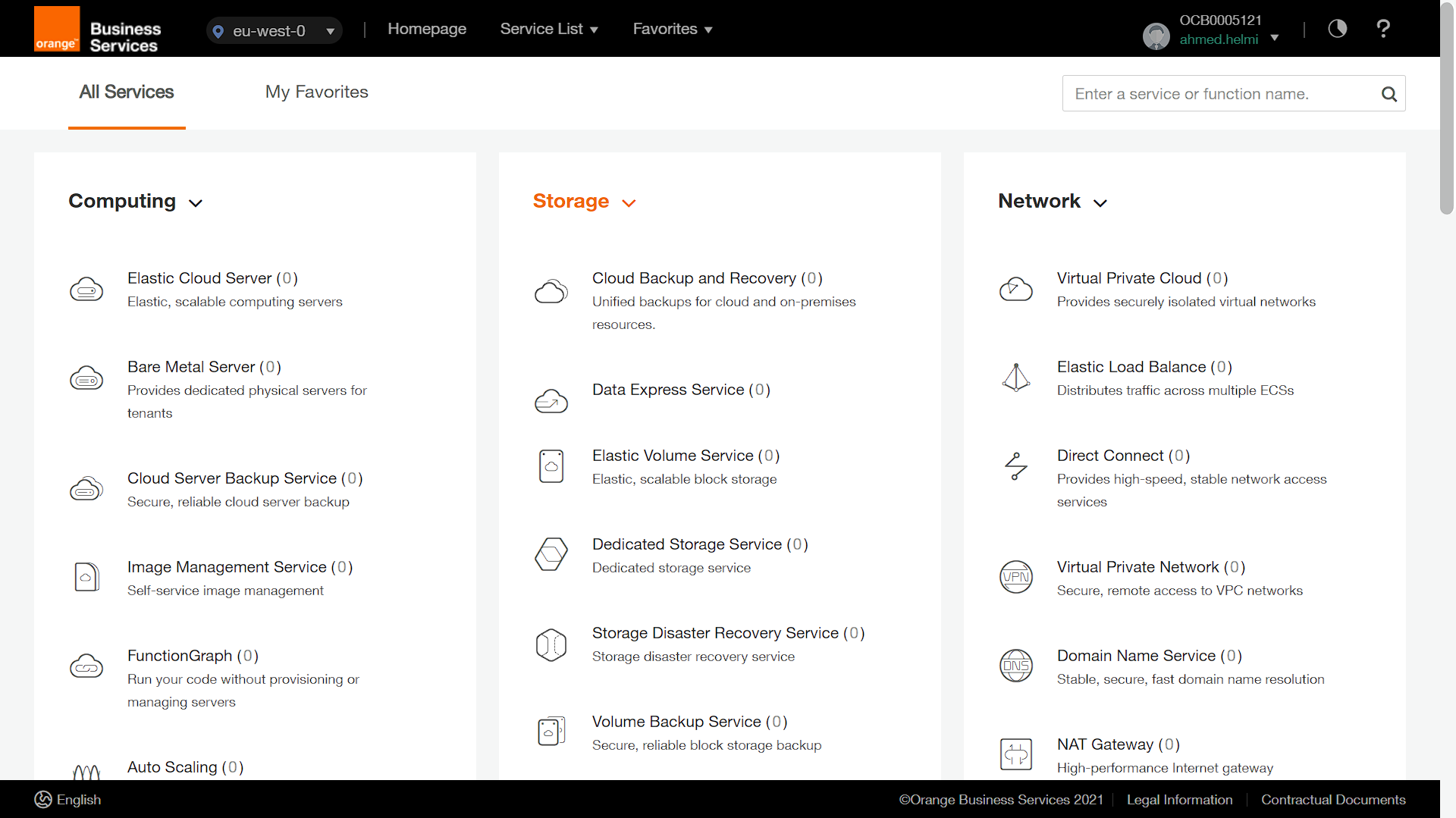
Git rm –cached <file>

Soft vs hard resets

# Flexible engine:

Flexible engine architecture.

Anti Affiniti ECS Group: To ensure that vms are created on different servers



FunctionGraph (Serverless)

# APIs:

SOAP RPC.

RPC (executes) Rest doesn’t(stateless)!

Postman variables scopes: Data<- Environment <- Collection <- global (smaller to bigger scope) (smaller applies when similar)

<https://postman-echo.com/get> echoes back what was sent

Current variable vs Initial Variable

# Terraform:

* Complex types.
* Image and Data.
* **Terraform state file**. To follow creation history. Has auto generated code. Auto generated locally. Backup extension looks like JSON. Somehow looks like logs. Has all the info about the already created. Shared on server. Tfstate can be uploaded to a bucket (best practice).
* Lifecycle should be automated by terraform and not manually. What initialized with terraform must be maintained only by terraform manual edits are not supported.
* One provider only
* Var application: in cmd, in tfvar
* Terraform.tfvars
* Data vs resource
* ***terraform state list***
* **Depens\_on=**
* For\_each
* Remote state file (S3) file
* Provisioners for software configuration management. \*\*\*\* ADVANCED \*\*\*\*

# Docker:

* LXC in Linux as alternative. Rocket & Podman.
* Images can exist without container but not the other way around.
* Containers created from images. (Images is the blueprint).
* Docker File Syntax Example:

FROM ${Image}

RUN <command>

* Bind vs volume vs tmpfs

# Ansible:

* Writing python scripts for ansible custom modules.
* Ansible can invoke terraform \*\* ADVANCED \*\*.
* Single master server side even handling errors can be mitigated. Creating replicas is not Important.
* CMDB is optional created & managed by user.
* API for ansible can be installed and used (Ansible Tower)
* Windows through WinRM.
* Client better has python installed to interpret client code.
* **INVENTORIES**:

/etc/ansible/hosts -> to configure machine.

[control] [<machine-name / all>]

Control ansible\_host=192.168.1.2 <machine-name> ansible\_host=<network address / DNS>

* Ad hoc command: - ansible <host/group/client> -m <module>/command -a ‘<Arguments>’ -i <inventory>
* Handlers. Notify. Do something once state matched.
* Ansible Tower Alternatively AWX.

# Kubernetes K8s:

* Kubernetes is an environment to manage clusters.
* A system not a tool.
* Fully reliant on Containerization.
* Round robin as a type of worker node processing.
* Cluster is a group of nodes or vms.

Features:

* Automated Scheduling
* Self-healing
* Auto rollouts
* Horizontal scaling and load balancing.

Kube apiserver (REST server to manage clusters)

Etcd datastore for cluster has the meta data of clusters. Very important to backup.

Kube controller manager

Kube scheduler (labeling)

\_\_\_\_

**Kube proxy**

Preparation to install:

Setup Bridge ip tables

Turn off swap mem. K8s don’t recognize it as memory.

# Gitlab CI/CD:

* Gitlab “.gitlab-ci.yml” file has the instructions to manage runners.
* Runner agent that run CI/CD jobs.
* Runner can run multiple jobs.
* Runner should have docker to manage tools through containers.
* **Executer**: is the environment that runner works on.
* By default, runner has one executer.
* Use tags to decide the executers for a runner.
* Docker as executer is the best practice.
* Pipelines are the top-level components.
* Jobs in the same stage run in parallel.
* A job MUST have script.
* **Global keywords :**
* only: only used to indicate running the job on certain branch.
* when: (manual, …) when manual means that it needs manual intervention.
* tags: (tags) list of tags to select runner.
* artifacts: file or directory we want to keep. Usually after build
* cache: file or directory we want to keep.
* before\_script: usually before commands that job runs.

Cache vs Artifact:

* Artifact usually built.
* Cache files can be stored and used (cant be reused).
* Artifact expire after 30 days by default.