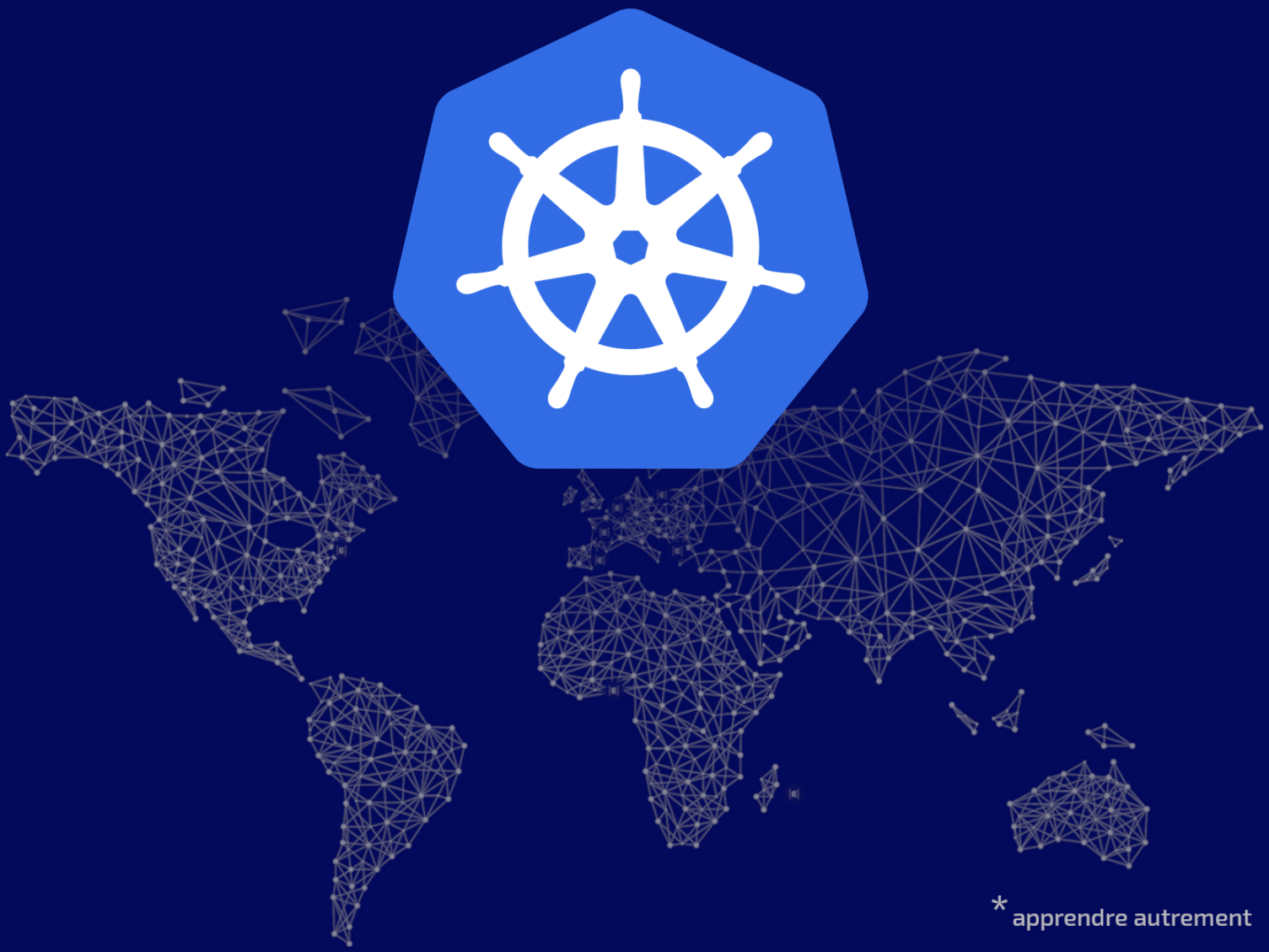
{EPITECH}
LEARN DIFFERENT*

KUBEQUEST

SAILING THROUGH THE CLOUDS

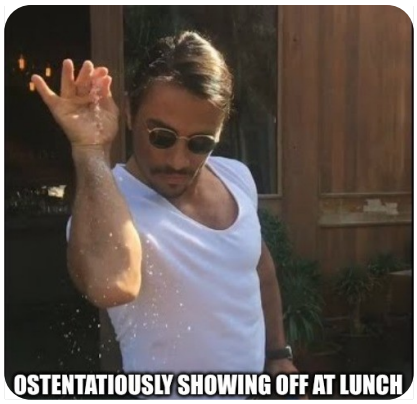


* apprendre autrement

KUBEQUEST

Welcome to an exhilarating journey into the world of cloud computing and containerization! You are part of a dynamic team at a forward-thinking company, poised to explore the robust capabilities of Kubernetes.

Your mission is to revolutionize the company's containerization infrastructure, showcasing your skills in software engineering and system management.



Objectives

Your primary task is to design and deploy a fully-equipped Kubernetes cluster, not only to host and present application, but also with management tools and administration components.

Below are the goals you need to achieve, with some *tool suggestions* as starting points :

- ✓ an internal **load balancer** to expose apps and services of your infra (*nginx-ingress*) ;
- ✓ a user-friendly **dashboard** to access and manage the containers (*kubernetes-dashboard*) ;
- ✓ a **monitoring stack** to collect and visualize system and apps metrics (*kube-prometheus*) ;
- ✓ a **GitOps** repository with all these reusable components manifests (*kustomize*) ;
- ✓ a **logging stack** to collect and visualize some logs (*loki*).

After that, convert and deploy an application which was actually deployed using *docker-compose*:

- ✓ create an **helm chart** for this application, and use official one for database (*helm*) ;
- ✓ create a **GitOps** repository to deploy this application (*kustomize*) ;
- ✓ **automate** deployment and verify application status.

You can find the application here : `https://gitlab.infra.connectwork.fr/epitech/sample-app.git`.

During this experimentation, implement all the “best practices” in a Kubernetes environment :

- ✓ define resources **limits and requests** on CPU and ram consumption ;
- ✓ use **secrets** to host sensible data ;
- ✓ **label** all your resources, following kubernetes recommandations ;
- ✓ add **redundancy** on your application, with multiple replicas and affinity rules, at least for “application” containers ;
- ✓ add **persistent storage** for your database, and implement usual backup system using Kubernetes.

You’ll also have to add security and control :

- ✓ add a **validating webhook** that can validate and control request (opa) ;
- ✓ add **authentification** to your Kubernetes API and your different tools (dex oauth-proxy).

Laboratory

Via the Azure Portal, you will be able to access to an Azure DevTest Labs environment, allowing you to deploy a specific number of virtual machines, services and resources:

- ✓ your deployment of the application and it's database ;
- ✓ monitoring tools.

You can only deploy to the Lab created in ResourceGroup, already present in portal.
You can use terraform: `https://registry.terraform.io/providers/hashicorp/azurerm/3.60.0/docs/resources/dev_test_linux_virtual_machine.html`

However, it's important to note that there are certain limits and restrictions in place to ensure efficient resource utilization and adherence to azure policies.

Because the resources deployed uses a live subscription, you MUST manage them responsibly. You'll have to ensure cost optimization and compliance by leveraging Infrastructure as Code (IaC) and configuration tools.

We strongly advise shutting down all virtual machines at the end of each day to conserve resources and minimize costs.

Delivery

Push your configurations into a repo.

Documentation is mandatory, especially the containers setup steps and commands.

Defense

Concerning the final defense of your project:

1. before presenting, start a fresh new Kubernetes cluster in the cloud, with many nodes ;
2. while presenting, deploy the services into the orchestrator using nothing more than `kubectl apply -f ... -f ... -f ...`, `kustomize...`, `helm ...` ;
3. in order to demonstrate some features, such as auto-scaling, execute some live scripts or commands you wrote beforehand in order to send lots of requests to your applications ;
4. demonstrate a full deployment process and a broken deployment with automatic rollback.

Feel free to enrich the applications code with some memory leaks, loops consuming CPU, or anything that could lead to errors and container failure.

Bonuses

You may also want to add some extras, such as:

- ✓ an encryption (with *Let's Encrypt* and *cert-manager*) ;
- ✓ *argoCD* to manage your components and GitOps repositories ;
- ✓ setup multi-tenant and authentification to *prometheus* and *loki*, via *kube-rbac-proxy* ;
- ✓ evaluate Helm artefacts security ;
- ✓ define the orchestrator access permissions ;
- ✓ reinforce the network access ;
- ✓ create some lightweight Docker images ;
- ✓ guarantee zero-downtime deployment (successful HTTP request during deployment) ;
- ✓ pull Docker images from a private registry ;
- ✓ …

v 0.1.4

* apprendre autrement