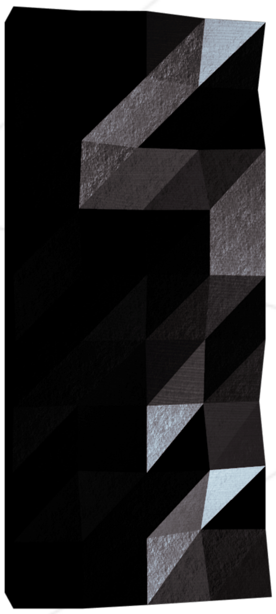

IaC-1

Infrastructure as Code

Charles Garoux chgaroux@student.42lyon.fr

Summary: architect a Cloud infrastructure and deploy it automatically with modern tools

Version: 1.0.0

Contents

Chapter I

Forewords

The yak or yak is a large species of domestic ruminant with a long fleece of the Himalayas. The female yak is called dri or drimo by the Tibetans and nak by the Sherpas.
Source : internet

In DevOps we say "cattle not pets", so in this image we have "pet" and "cattle". Hint: the iac is your buddy.

Chapter II

Introduction

This is the first of a series of projects designed to introduce the DevOps technique of Infrastructure as Code (IaC).

You're going to be the architect and magician for this project.
You'll architect the Cloud infrastructure for a provided web application, and you'll make magic when your infrastructure automatically deploys, leaving you time to grab a coffee.

Chapter III

Scenario

You're in a company, you're the developer and you've got a boss who has no sense of priorities.

One morning, the boss shows up in front of your office with a coffee in one hand and a USB stick in the other. A conversation begins:

- The boss starts: "My nephew has coded something for his internship", he hands over the USB key and continues: "Put it in the cloud and it needs to be robust for when there's traffic."
- You begin to reply: "But..."
- The boss cuts you off "Perfect, you've got 3 months", and puts the key on your desk.

No sooner had you looked at the USB key than he disappeared from the open space.

When you open the USB key, you'll find a classic web application with basic, but existing, documentation.

Chapter IV

Platform choice

The school does not provide the servers required for the project. You'll have to provide your own infrastructure.

Fortunately, most Cloud providers offer trials in the form of credits or services with free quotas. These trials can make it possible to do the project at a lower cost.
With sensible and smart usage, you can get by for just a few euros. As a reminder, "cattle not pets", shut down your servers when you're not working.

`The author of the project has done so at a cost of less than 5 euros.`

There are many different Cloud platforms. Here is a non-exhaustive list of well-known platforms:

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure

The choice of Cloud provider platform is yours and this decision is part of the project, so in correction you'll explain your choice of platform.

`This freedom is limited to solving the subject's problems without cheating.`

Chapter V

Choice of IaC technology

In addition to the platform, this project gives you the choice of deployment technology in Infrastructure as Code.

Different tools exist, some are proprietary to Cloud platforms for deploying their services:

- CloudFormation d'Amazon Web Service
- Azure Resource Manager de Microsoft Azure
- Cloud Deployment Manager de Google Cloud

Other tools can deploy across different providers:

- Terraform by HashiCorp
- Pulumi by Pulumi

The tools presented here are well-known examples, so it's up to you to choose the one you want to use. Make sure that the technology you choose matches your subject's requirements.

`Terraform and Pulumi will surely match the subject.`

The choice of technology platform is yours, and this decision is part of the project, so in correction you'll explain your choice of technology.

This freedom is limited to solving the subject's problems without cheating.

Chapter VI

Mandatory part

You must be able to deploy a Cloud infrastructure to host the application provided as a resource of the project.

Cloud infrastructure is provisioned using an Infrastructure As Code (IaC) tool.

VI.1 Cloud

The application runs continuously on at least 2 separate servers, as far as the supplier allows, preferably on different server farms.
Make it visible in the application which server is serving the visible page (e.g. display the server's IP address).
It is permitted to modify the source code of the supplied application to meet this need.

Set up a system to distribute the load evenly between servers.

The user should be able to stay connected for a normal session duration, even if they refresh the page.

Any addition, modification or deletion made on the application must be immediately applied to the other instances. A delay of a few seconds is acceptable, but will require justification.

The infrastructure must be able to withstand various failures, such as the loss of a server instance or high load (CPU/RAM), to ensure the high availability of the application.
We recommend automating high-availability tests using a script, tool or other appropriate solution.

The infrastructure must be able to handle the load by adjusting the number of server instances according to their capacity to support this load.
We recommend automating scalability tests using a script, tool or other appropriate solution.

Costs need to be optimized as best you can, bearing in mind that you'll need to justify your choice of Cloud services and options.
It's up to you to balance cost savings and infrastructure capacity.
It is advisable to provide a configuration option offering minimum cost, in order to limit development expenses.

Security measures must be put in place to prevent access to resources or services that should not be publicly accessible. Even in the Cloud, usual security rules must be adhered to.
Secrets must be protected in the best possible way. There should be no secrets stored in the code repository, scripts or unsecured files.

The administrator must be alerted in the event of malfunctions, such as application inaccessibility or unresponsive servers.

VI.2 Infrastructure as Code

The code should be as modular and reusable as possible.

The deployment region must be easy to select, for example by being defined at the root of the configuration.
This selection should be possible with simplified naming, for example using names like "EU" or "Paris" rather than identifiers like "eu-west-3".

The capacity of server and database instances must be easy to select, for example by being defined at the root of the configuration.
It should be possible to make this selection using simplified naming, for example by using terms such as "small", "medium" or "large" rather than technical identifiers such as "e2-standard-1".

Global static values are modifiable by the end user to meet their needs. They should be accessible without requiring modifications to the deployment code, for example for the selectable server type, the email address for alerts, etc.
The file format used to store them is free, as long as it is compatible with the technology used.

VI.3 Documentation & accessibility

One or more schemas must be created to describe the structure of the infrastructure.
The infrastructure may vary depending on the selections made during deployment, such as the number of server farms, for example. In this context, it is acceptable to provide a schema representing the default infrastructure.

The application and infrastructure should be deployable with as few actions and as little technical knowledge as possible.

This implies documenting the few steps necessary for deployment, such as the commands to execute, retrieving secrets, installing secrets, etc.

The application must be operational after deployment, with no further action required.

Chapter VII

Bonus

There are lots of possibilities, so have fun exploring and experimenting!

Here are some ideas:

- Deploy the application in multiple regions around the world, with content synchronization between regions
- Integrate Chaos Engineering to test infrastructure resilience
- Deploy a set of relevant alarms and metrics grouped together on a dashboard
- Add a domain name and enable HTTPS protocol to access the application
- Add log management functionality (the application already produces logs)
- Add an automatic database backup and restore system
- Create carefull documentation using a dedicated tool, such as Docusaurus for example

Chapter VIII

Elimination case

It is strictly prohibited to have one or more secrets (passwords, API keys, etc.) in plain text in the repository. Furthermore, it is strictly forbidden to include them in a script or file that would be accessible after the deployment is completed.

Chapter IX

Forbidden

Elements that hinder the pedagogical dynamics of the project are strictly forbidden. Here are some examples of platforms, services and technologies that are prohibited:

- PaaS : Vercel, Google App Engine, Amazon Elastic Beanstalk, Azure App Service, ...
- Serverless : AWS Lambda, Google Cloud Functions, Azure Functions, ...
- Orchestrators (managed or not) managing server clusters: Kubernetes, Docker Swarm, Nomad, OpenShift, ...

Although orchestrators are prohibited, the use of Docker Compose is allowed, as it cannot manage server clusters.

The use of external modules to deploy an infrastructure is prohibited. For example, using a Terraform module to deploy a complete network on AWS from the Terraform Registry or Github is prohibited.

Modifying the provided application with the intention of deceiving the evaluator by displaying false information is obviously prohibited.

Any attempt to defy these prohibitions will result in a -42 to the project.

`If you feel that something compromises the pedagogical dynamic, it is probably forbidden. If in doubt, please contact the educational team and/or the project author.`

Chapter X

Return and peer-evaluation

The assignment is entirely submitted via a Git repository which must contain:

- The code used for deployment
- The modified application
- At least basic documentation so that the evaluator can deploy it themselves
- At least one diagram describing the infrastructure
- Everything needed for the project and the correction