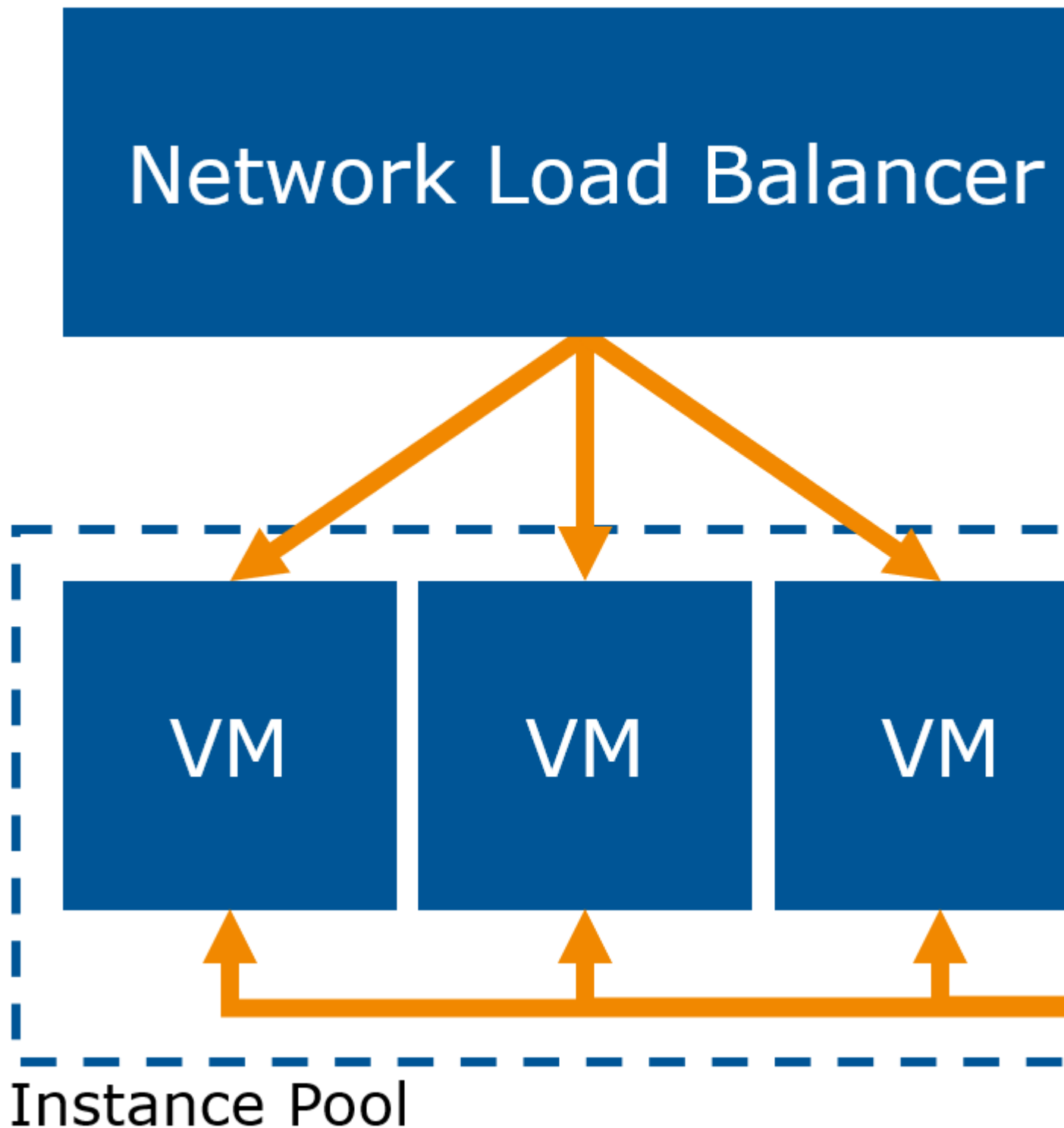


## Project work

You are the cloud architect for a small work-for-hire company. A client wants to hire you but they are sceptical about your ability to design a system that can scale. Your job is to design a system that can scale. Auto scaling management should automatically launch new cloud servers when the load is high and remove servers when demand is low.



### Warning

Do not continuously run your setup on Exoscale or you will run out of budget! Budget limitations are part of building cloud systems.

After taking a look at the capabilities of the cloud provider and discussing the constraints with your colleagues you decide that:

- You are going to use [Terraform](#) to automate the setup and tear down of the cloud infrastructure. This is necessary because of the constraints.
- You will use [instance pools](#) to manage the variable number of cloud servers and [Network Load Balancers](#) to balance the load.
- You will set up a dedicated monitoring and management instance which will run [Prometheus](#) to automatically monitor the system and alert you when it starts to consume too much.
- On the instance pool you will deploy the [Prometheus node exporter](#) to monitor CPU usage.
- You will install [Grafana](#) to provide a monitoring dashboard and the ability to send webhooks.
- You will configure an alert webhook in Grafana that sends a webhook to an application written by you. If the average CPU usage is too high, the application will scale the instance pool up or down.
- You will write an application that receives this webhook and every 60 seconds scales the instance pool up or down if a webhook is received.

As you also have to demonstrate to the client that you can work in an agile methodology you agree in 4 week sprints with a 2-week demo cycle.

As a dummy service to generate load you will use [http-load-generator](#).

The manual way to implement this architecture is described [on the Exoscale blog](#). We recommend reading and doing the steps manually.

Optionally, you can make use of the following (incurs a 5% point-penalty each):

- [prometheus-sd-exoscale-instance-pools](#) to feed instance pool data into Prometheus
- [exoscale-grafana-autoscaler](#) to drive the autoscaling behavior.

## Handing in your project work

In order to hand in your project work you must upload your code to a Git repository, for example on GitHub. Submit the link to the repository.

Your Terraform code **must** be able to install the complete infrastructure into an empty Exoscale account. It **must** ask for the Exoscale API key and the Exoscale account ID.

If you opt to only implement the manual method (without Terraform) please enter `manual` in instead of the text field and we will not evaluate your solution.

### Warning

Keep in mind that only submitting a manual solution will give you minimal points. We strongly recommend you ask for help if you are stuck.

### Note

The automated version has to run without manual intervention beyond inserting variables. Additionally, the solutions will be spot-checked.

## Acceptance criteria

The system will be tested with Apache Benchmark using the following command line:

```
ab -c 2 -n 1000 http://your-nlb-ip/load
```

Your system must be able to launch enough instances within 10 minutes to get the average load back under 80%. When AE

## Getting help

Please see the [getting help page](#).