Create a Google Kubernetes Engine (GKE) cluster with jx

In the lesson we will discuss how to create a GKE cluster with jx.

We'll cover the following

- GCP project
- Storing project name in an environment variable
- Creating a Google Kubernetes Engine (GKE) cluster with jx
  - Cluster name and project ID
  - Region and machine type
  - Maximum and minimum number of nodes
  - Default admin password
  - Default environment prefix
- What do we get from this command?
  - GCP authentication
  - GCP services
  - Cluster creation
  - Jenkins X and other installations
  - Git credentials
  - Ingress installation
  - Custom domain name
  - Long-term logs and Google Cloud Zone
  - Git and GitHub settings
  - Organization
  - GitOps
  - kubectl context change

GCP project

Everything we do inside a **Google Cloud Platform (GCP)** is inside a project. That includes GKE clusters. If we are to let `jx` create a cluster for us, we need to know the name of the GCP project where we'll put it. If you do not have a project you'd like to use, please visit the Manage Resources page to create a new one. Make sure to enable billing for that project.

Storing project name in an environment variable

No matter whether you created a new project specifically for Jenkins X or chose to reuse one that you already have, we'll need to know its name. To simplify the process, we'll store it in an environment variable.

Please make sure to replace `[...]` with the name of the GCP project before executing the command that follows.

```
PROJECT=[...]
```

Creating a Google Kubernetes Engine (GKE) cluster with `jx`

Now we're ready to create a GKE cluster with all the tools installed and configured.

Cluster name and project ID

We'll name it `jx-rocks` (`-n`) and let it reside inside the project we just defined (`-p`).

Region and machine type

It'll run inside `us-east1` region (`-r`) and on `n1-standard-2` (2 CPUs and 7.5 GB RAM) machines (`-m`). Feel free to reduce that to `n1-standard-1` if you're concerned about the cost. Since GKE auto-scales nodes, the cluster will scale up if we need more.

Maximum and minimum number of nodes

While on the subject of scaling, we'll have a minimum of three nodes (`--min-num-nodes`) and we'll cap it at five (`--max-num-nodes`).

Default admin password

We'll also set the default Jenkins X password to `admin` (`--default-admin-password`).

Otherwise, the process will create a random one.

Default environment prefix

Finally, we'll set `jx-rocks` as the default environment prefix (`--default-environment-prefix`). A part of the process will create two GitHub repositories, one for staging and the other for the production environment.

The `jx-rocks` prefix will be used to form their names. We won't go into much detail about those environments and repositories just yet. That's reserved for one of the following chapters.

Feel free to change any of the values in the command that follows to better suit your needs. Or, keep them as they are. After all, this is only a practice, and you'll be able to destroy the cluster and recreate it later with different values.

```
jx create cluster gke \
    --cluster-name jx-rocks \
    --project-id $PROJECT \
    --region us-east1 \
    --machine-type n1-standard-2 \
    --min-num-nodes 1 \
    --max-num-nodes 2 \
    --default-admin-password admin \
    --default-environment-prefix jx-rocks
```

What do we get from this command?

Let's explore what we're getting with this command. You should be able to correlate my explanation with the console output.

GCP authentication

First, the GCP authentication screen should open asking you to confirm that you are indeed who you claim you are. If that does not happen, please open the link provided in the output manually.

GCP services

Next, `jx` will ensure that all the GCP services we need (`container` and `compute`) are enabled.

Cluster creation

Once we're authenticated and the services are enabled, `jx` will create a cluster

after you answer a couple of questions. The default values should suffice for now. It should take only a few minutes.

Once the GKE cluster is up and running, the process will create a `jx` Namespace. It will also modify your local `kubectl` context and create a ClusterRoleBinding that will give you the necessary administrative permissions.

Jenkins X and other installations

Next, the installation of Jenkins X itself and a few other applications (e.g., ChartMuseum for storing Helm charts) will start. The exact list of apps that will be installed depends on the Kubernetes flavor, the type of setup, and the hosting vendor. But, before it proceeds, it'll need to ask us a few other questions; What kind do we want to install? Static or serverless? Please answer with `Serverless Jenkins X Pipelines with Tekton`. Even though Jenkins X started its history with Jenkins, the preferred pipeline engine is Tekton, which is available through the serverless flavor of Jenkins X. We'll discuss Tekton and why Jenkins X is using it later.

Git credentials

At this point, `jx` will try to deduce your Git name and email. If it fails to do so, it'll ask you for that info.

Ingress installation

The next in line is Ingress. The process will try to find it inside the `kube-system` Namespace and install it if it's not there. The process installs it through a Helm chart. As a result, Ingress will create a load balancer that will provide an entry point into the cluster. This is the step that could fail during our setup. The GCP default quotas are very low, and you might not be allowed to create additional load balancers. If that's the case, please open the Quotas page, select those that are at the maximum, and click the *Edit Quotas* button. While increasing quota is a manual process, they do it relatively fast, so you should not have to wait very long.

Once the load balancer is created, `jx` will use its hostname to deduce the IP.

Custom domain name

Since we did not specify a custom domain for our cluster, the process will combine the IP of the load balancer with the nip.io service to create a fully qualified domain, and we'll be asked whether we want to proceed using it. Type `y` or merely

press the enter key to continue.

Long-term logs and Google Cloud Zone

Jenkins X will *enable long-term logs storage* automatically and ask you for the *Google Cloud Zone*. Feel free to keep the one selected by default.

Git and GitHub settings

Next, we'll be asked a few questions related to Git and GitHub. In most cases, all you have to do is confirm the suggested answer by pressing the enter key. As a result, `jx` will store the credentials internally so that it can continue interacting with GitHub on our behalf. It will also install the software necessary for the correct functioning of those environments (Namespaces) inside our cluster.

Organization

We're almost done! Only one question is pending. You'll be asked to `select the organization where you want to create the environment repository`, choose one from the list.

GitOps

The process will create two GitHub repositories; `environment-jx-rocks-staging` that describes the staging environment and `environment-jx-rocks-production` for production. Those repositories will hold the definitions of those environments. For example, when you decide to promote a release to production, your pipelines will not install anything directly. Instead, they will push changes to `environment-jx-rocks-production` which will, in turn, trigger another job that will comply with the updated definition of the environment.

That's GitOps

Nothing is done without recording a change in Git. Of course, for that process to work, we need new jobs in Jenkins, so the installation process created two jobs that correspond to those repositories. We'll discuss the environments in greater detail later.

`kubectl` context change

Finally, the `kubectl` context was changed to point to the `jx` Namespace, instead of `default`.

We'll get back to the new cluster and the tools that were installed and configured in the What Did We Get? section. Feel free to jump there if you have no interest in other Cloud providers or how to install Jenkins X inside an existing cluster.

Next up is the EKS.