Birkbeck
UNIVERSITY OF LONDON

Cloud Computing Workshop
Session 3: Kubernetes

Dimitris Kargatzis

Stelios Sotiriadis

Plan for today

- Lecture part 1:
  - What is an orchestrator?
- Lecture part 2:
  - Why do we need to learn about Kubernetes?
  - What is Kubernetes?
- Lab session:
  - What is the Google Kubernetes Engine (GKE)?
  - GKE Deployment lifecycle
  - Deploy a Kubernetes cluster in GCP!
  - Use Docker to create images and Kubernetes to deploy!

Orchestrators

What is orchestration?

- Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services.
  - Orchestration helps IT to more easily manage complex tasks and workflows.
- IT teams must manage many servers and applications but doing so manually isn't a scalable strategy.
  - The more complex an IT system, the more complex managing all the moving parts can become.

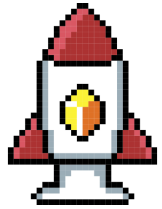
What is automation?

- Automation helps make your business more efficient by reducing or replacing human interaction with IT systems and instead using software to perform tasks
  - Reduce cost, complexity, and errors

Orchestration and Automation on the Cloud

- Automation refers to automating a single task
  - Push our code to Git
  - Create a container using Dockerfiles
- Orchestration refers to automating processes (workflow) involving many tasks
  - Create 100 containers and a load balancer pressing an enter!
  - Update my software:
    - From my laptop to production in less than 5 minutes
  - For orchestration we use Kubernetes!

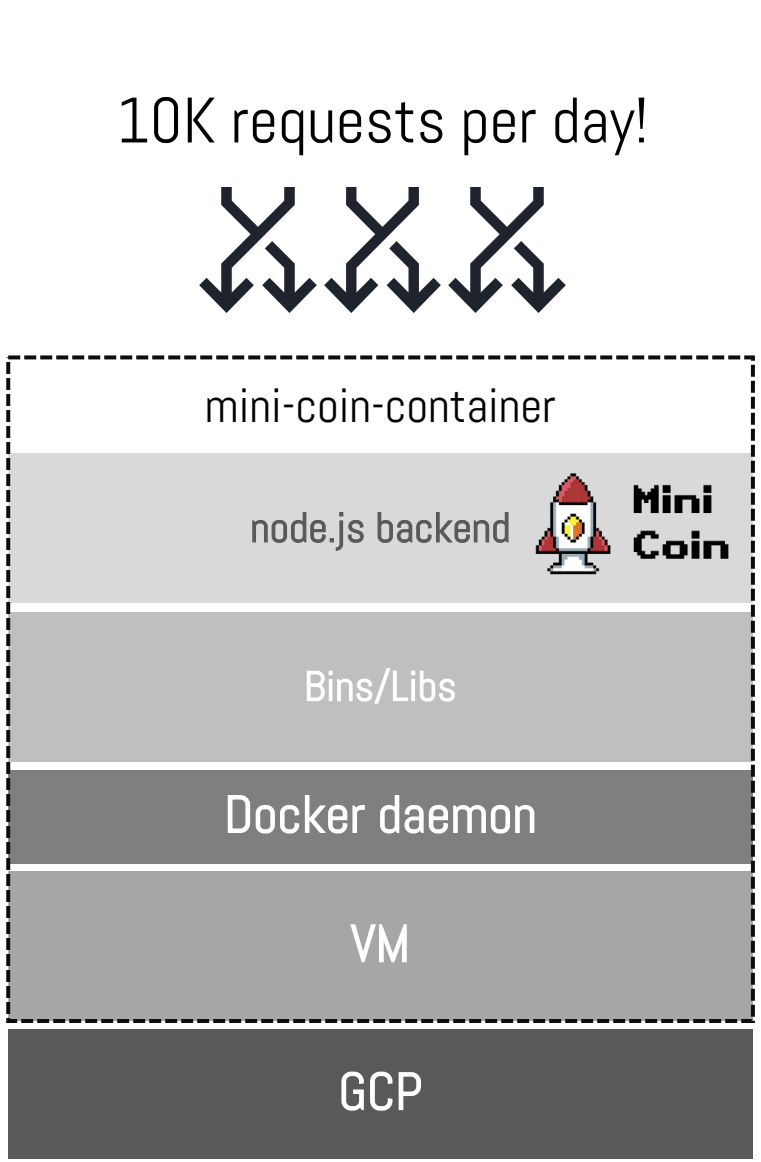
Why do we need Kubernetes?

Building the  Mini Coin

- We developed a **MiniCoin** app and we deployed on GCP in a container!
  - MiniCoin is a pay-per-use API to provide latest crypto prices
  - Btw, do you have a good idea about an API?
    - Check this link.
- Our API service is getting popular, and we are doing great!
- We move from thousand to tens of thousands of users!

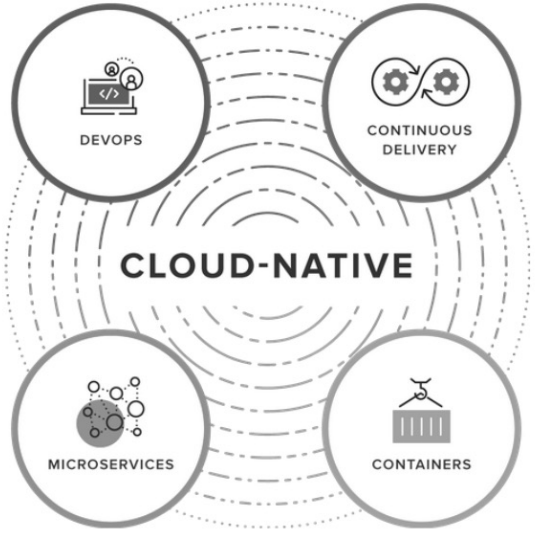
How are app looks like?

- Traffic is increased!
  - Good for business
- But the VM host is weak, maybe more resources are needed!
- Host crashed a couple of times ☹
- App is getting slow, latency is increased!
  - From milliseconds to seconds
- What can we do?
  - Add a second host!



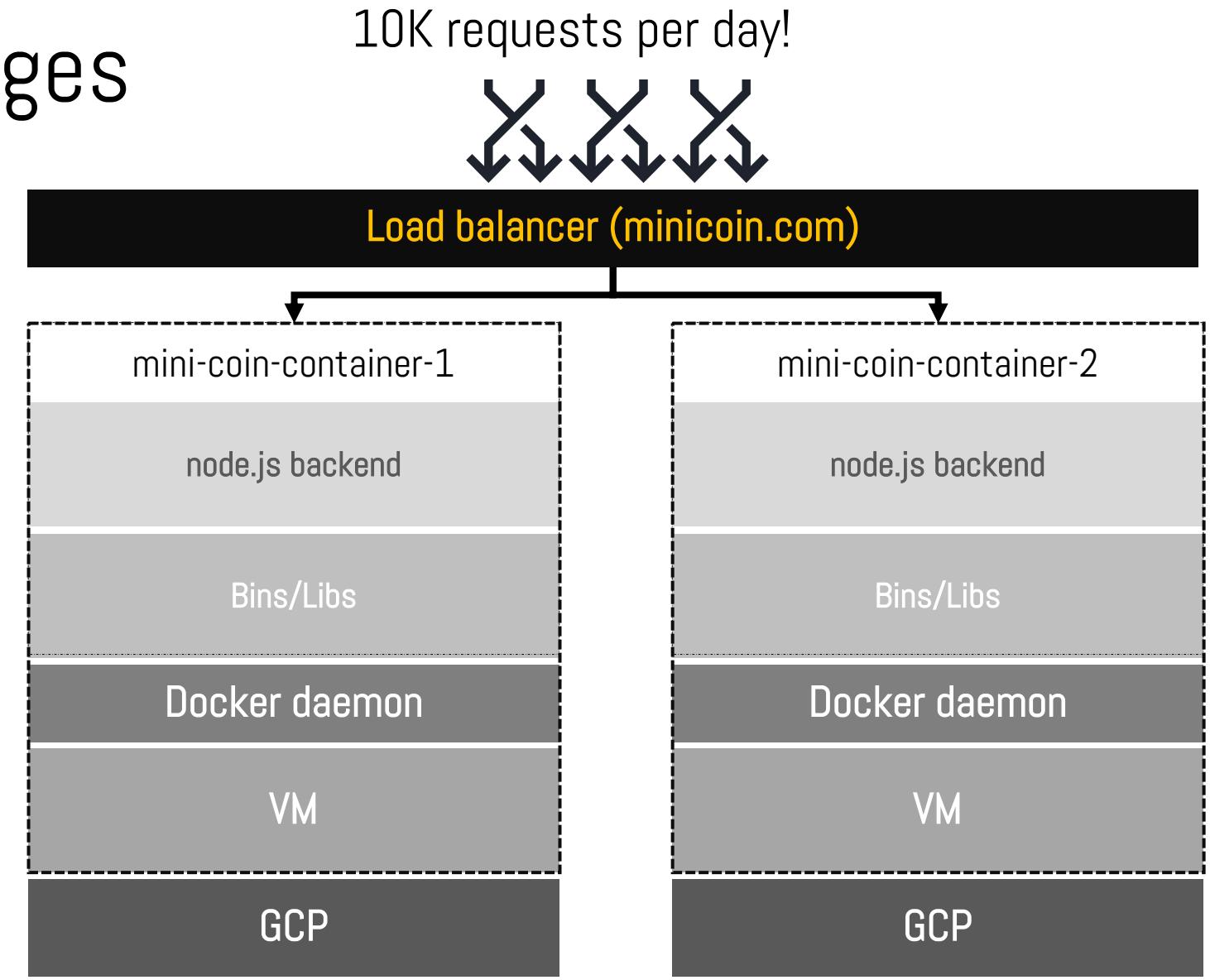
Cloud Native app

- A cloud-native application is a program that is designed for a cloud computing architecture.
- Briefly an app that is designed and developed considering cloud characteristics:
  - For example, to support any kind of demand, scale as necessary, be secure, survive failures etc.



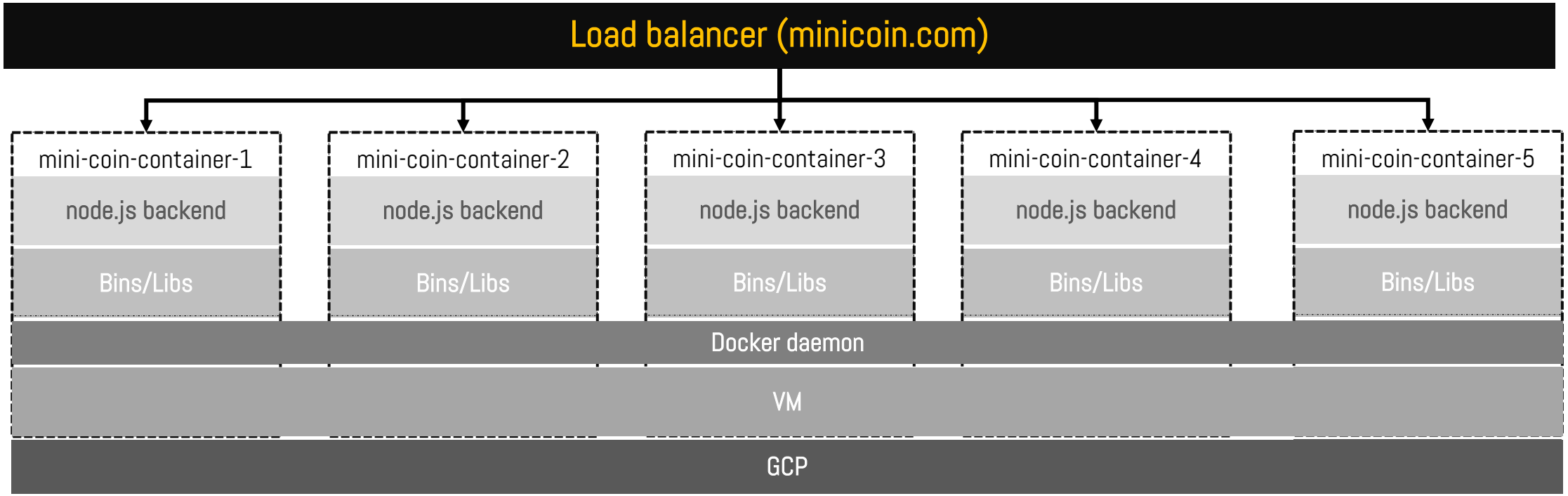
Manage traffic/outages

- We use both servers ☺
- If one goes down the other is serving
- Latency is back to normal!
- But… more and more people use our service!
  - We reach 1M users per day!
  - What should we do?



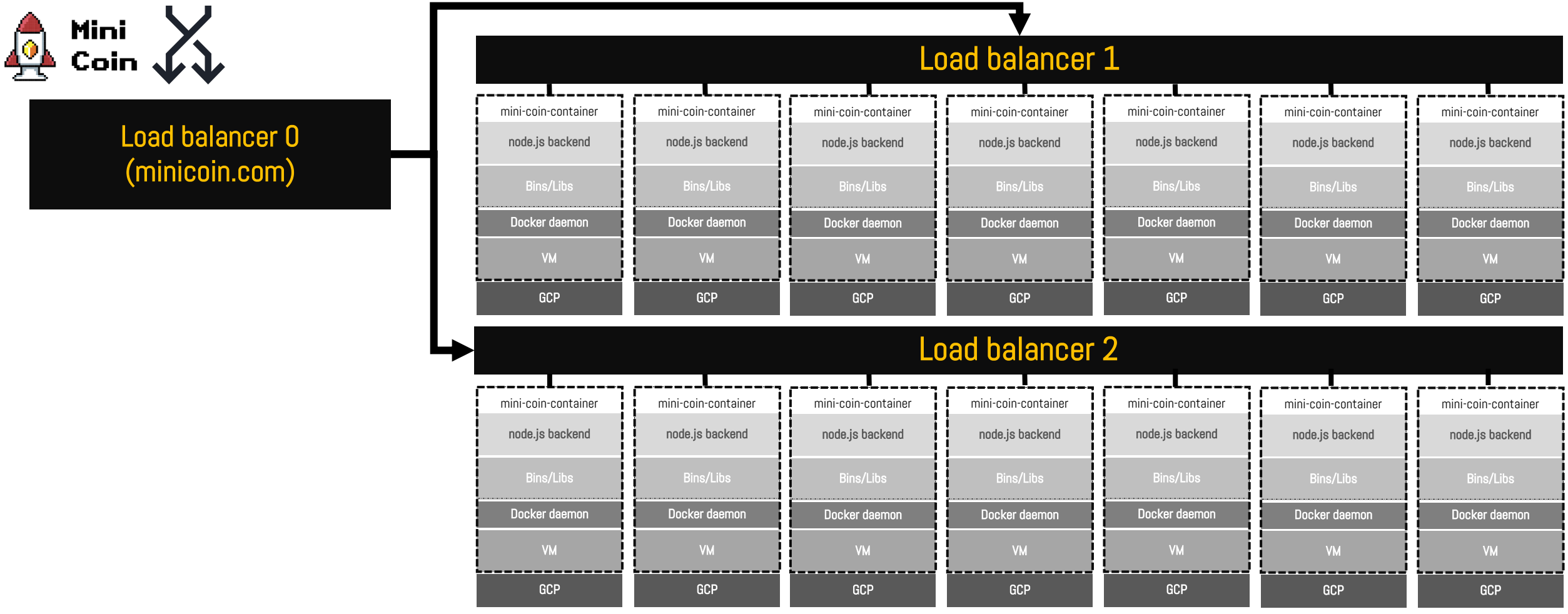
We are doing great! Let's add more...

1M request per day!



Again, we are doing great!

- Too great... now we need to serve 2M users!
- Let's add more!

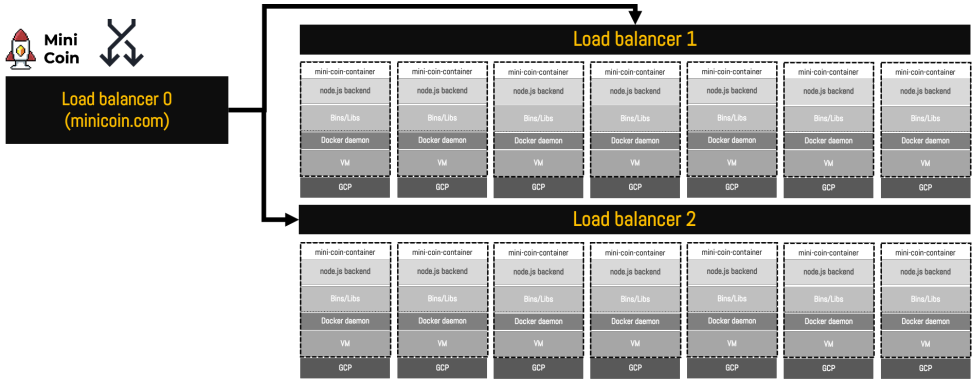


Now it's hard to manage...

- What if I have 100M users and I need to deploy thousands of containers?
  - Btw: Google starts over 2 billion containers per week, over 3000 started per second.
- Should I add thousands of containers?
- I can still do it with Docker and some custom scripts...!
  - But how can I manage this?
  - I need to hire an Ops guy!

Oh no!

- A new crypto coin is now out! We need to update our API!
- What can I do?
  - I scale so much that it's hard to manage!
  - I will need to hire an Ops guy!
  - Or a solution to automate this?
    - Better: a solution to orchestrate my infrastructure?
    - I need Kubernetes! ☺



What is Kubernetes?

- In 2014, Google released Kubernetes as an opens source software!
  - They donated Kubernetes to the Cloud Native Computing Foundation
    - … that serves as the vendor-neutral home for many of the fastest-growing open-source projects, including Kubernetes, Prometheus, and Envoy.
  - Google used containers in scale long before Docker for their services:
    - Search and Gmail

Kubernetes comes from the Greek word kevernitis (κυβερνήτης) that means "helmsman or pilot".

There you go!

Kubernetes

- Kubernetes aka K8s
  - It is an open-source system for automating deployment, scaling, and management of containerized applications
- Features:
  - Planet scale deployments
    - Google deploy billions of containers per week!
  - Flexibility
    - It grows with you to deliver your applications consistently and easily no matter how complex your needs are

Kubernetes & Docker

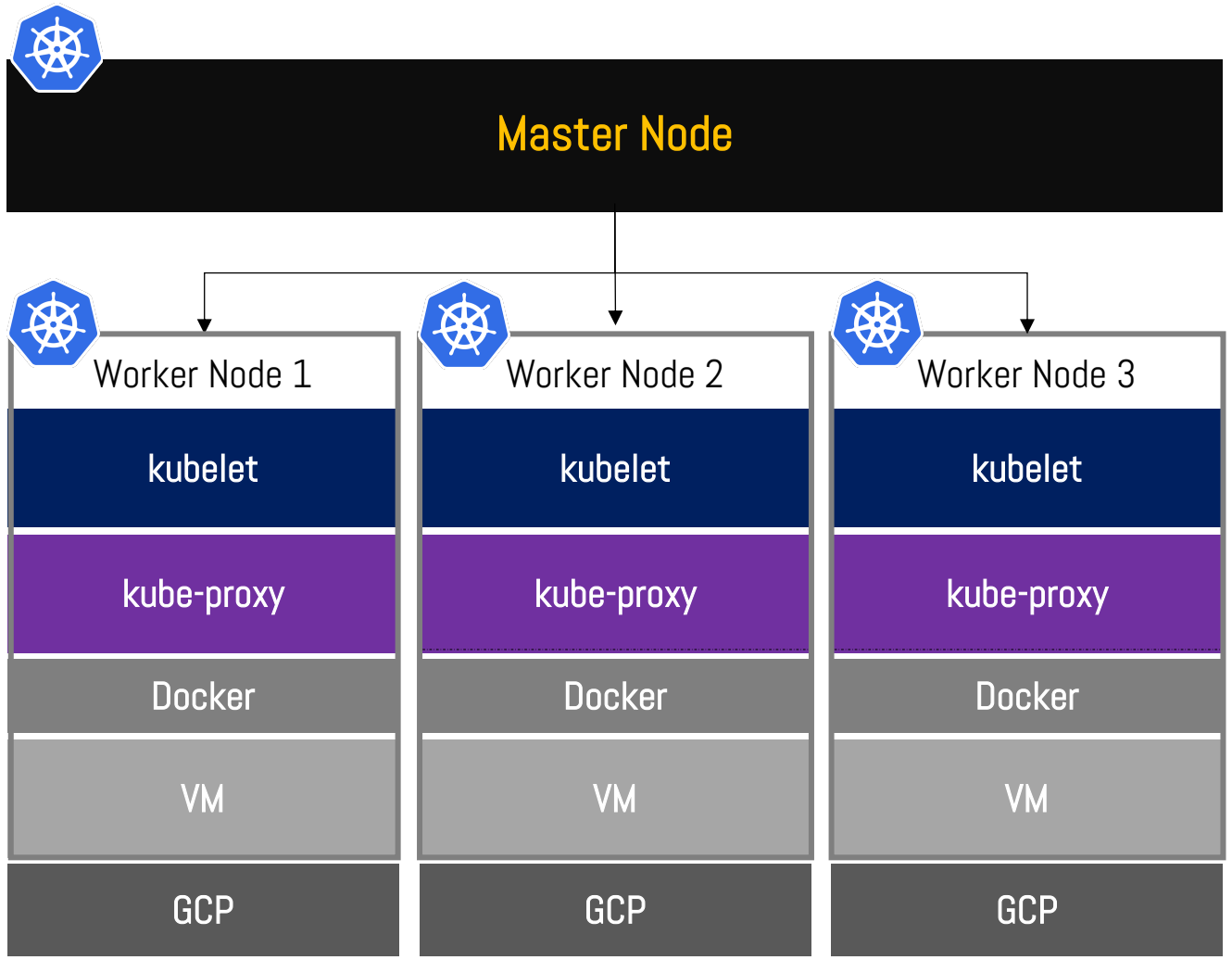
- Docker is amazing! It's fast, efficient and easy to use
  - Automation using Dockerfiles and DockerHub makes it easy to set up and use
- It can help us automate processes e.g., from my laptop to production, but we need something more!
  - Something to help Docker run mundane tasks and orchestrate deployments
  - We need to change our software architecture to support Kubernetes
    - Kubernetes and Docker work together

Why does Docker not orchestrate?

- It does!
  - It's called Docker Swarm
    - Docker swarm is **a container orchestration tool**, meaning that it allows the user to manage multiple containers deployed across multiple host machines.
- 2016-2017 we had the **ORCHESTRATOR WARS**
  - Docker Swarm, Mesosphere and Kubernetes competed
- Long story short:
  - K8S won!

The Kubernetes system

- The master node(s) host the control plane aspects of the cluster
  - Kubernetes API server
  - Scheduler
  - Controller
  - Etcd
- Workers are the workforce, they run our containers!

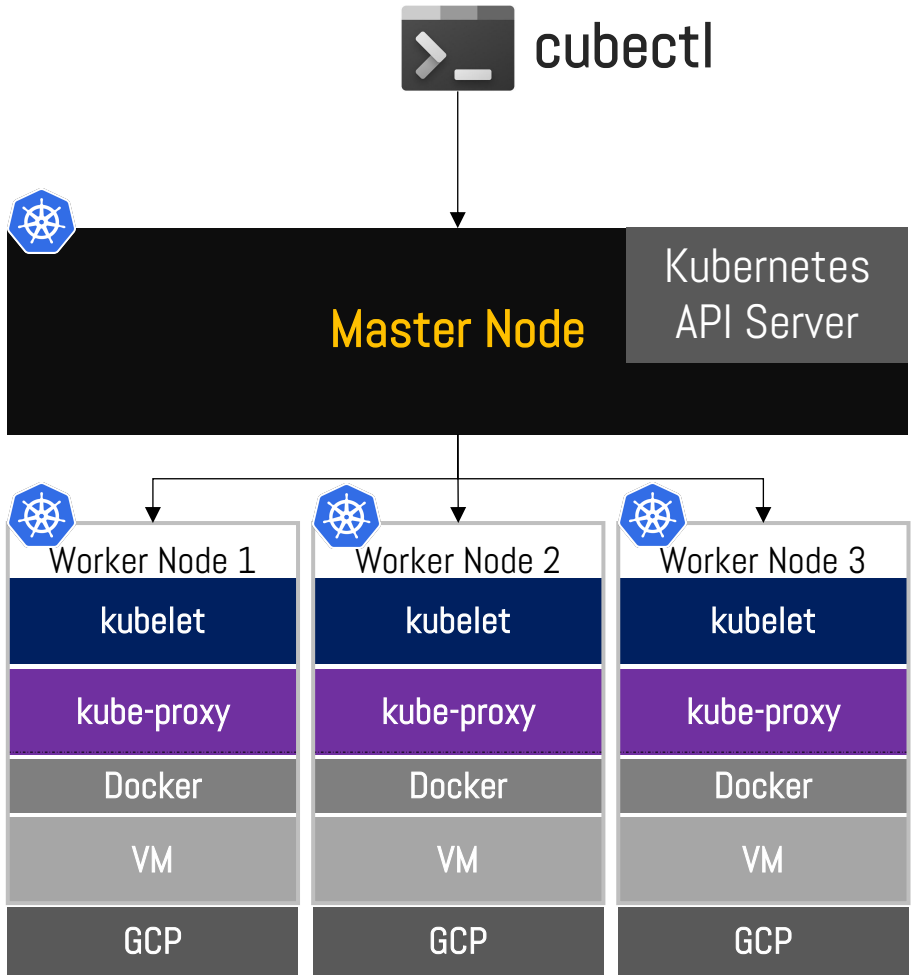


kubelet vs kube-proxy

- **Kubelet (or we can call it node)**
  - An **agent** that communicates with the master
  - Kubelet is the technology that applies, creates, updates, and destroys containers on a Kubernetes node
  - Each node runs a kubelet
    - Node registers in the API using CPU, memory and storage
  - Container runtime: Docker
- **kube-proxy**
  - A network proxy that runs on each node in your cluster
  - It maintains network rules on nodes

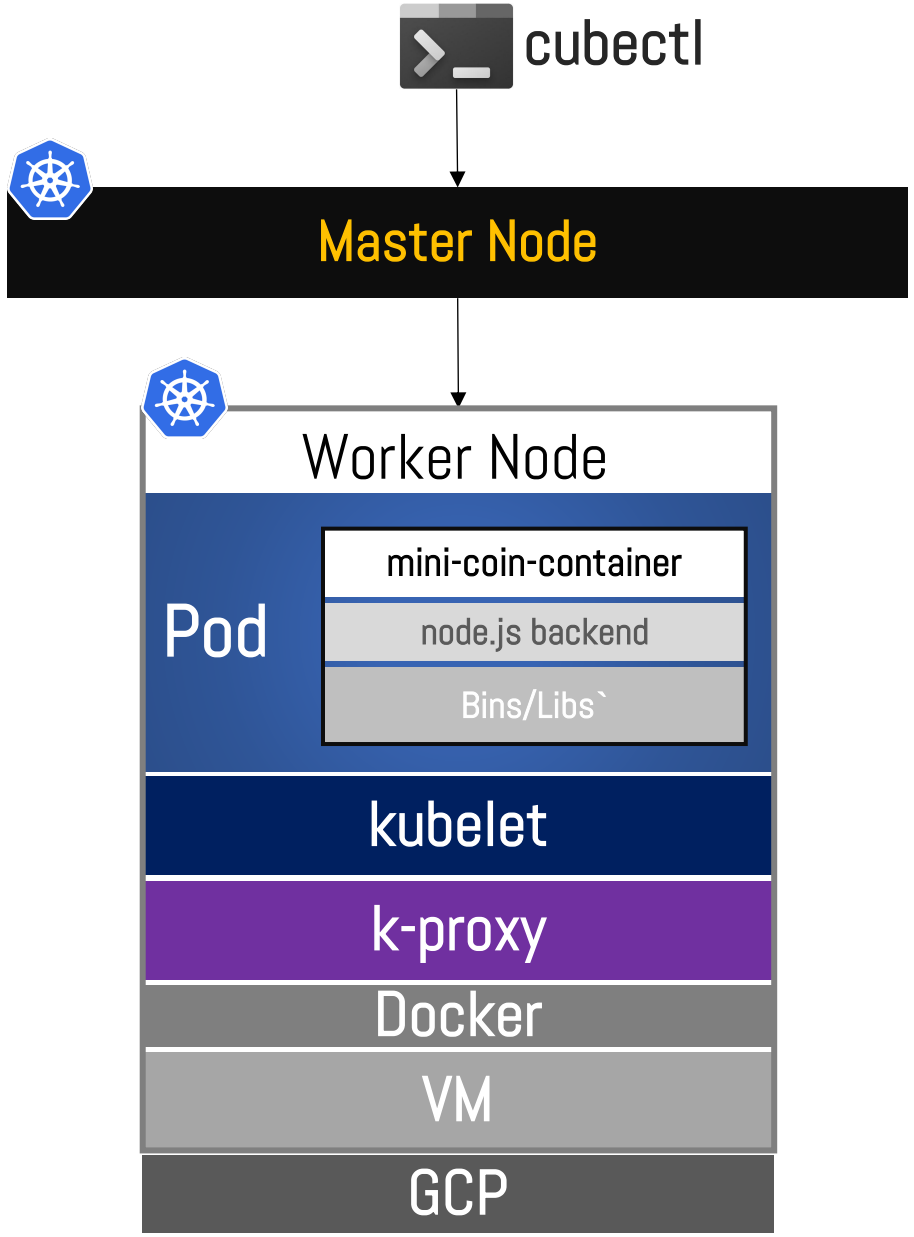
Kubernetes API Server

- The Kubernetes API server **validates and configures data for the API objects:**
  - Pods and services
- We use the API server to connect to the Master and run commands
- To communicate with the Kubernetes API Server we use a tool called **kubectl** (pronounced as kube-cattle)
  - Command line tool ☺



Kubernetes Pods

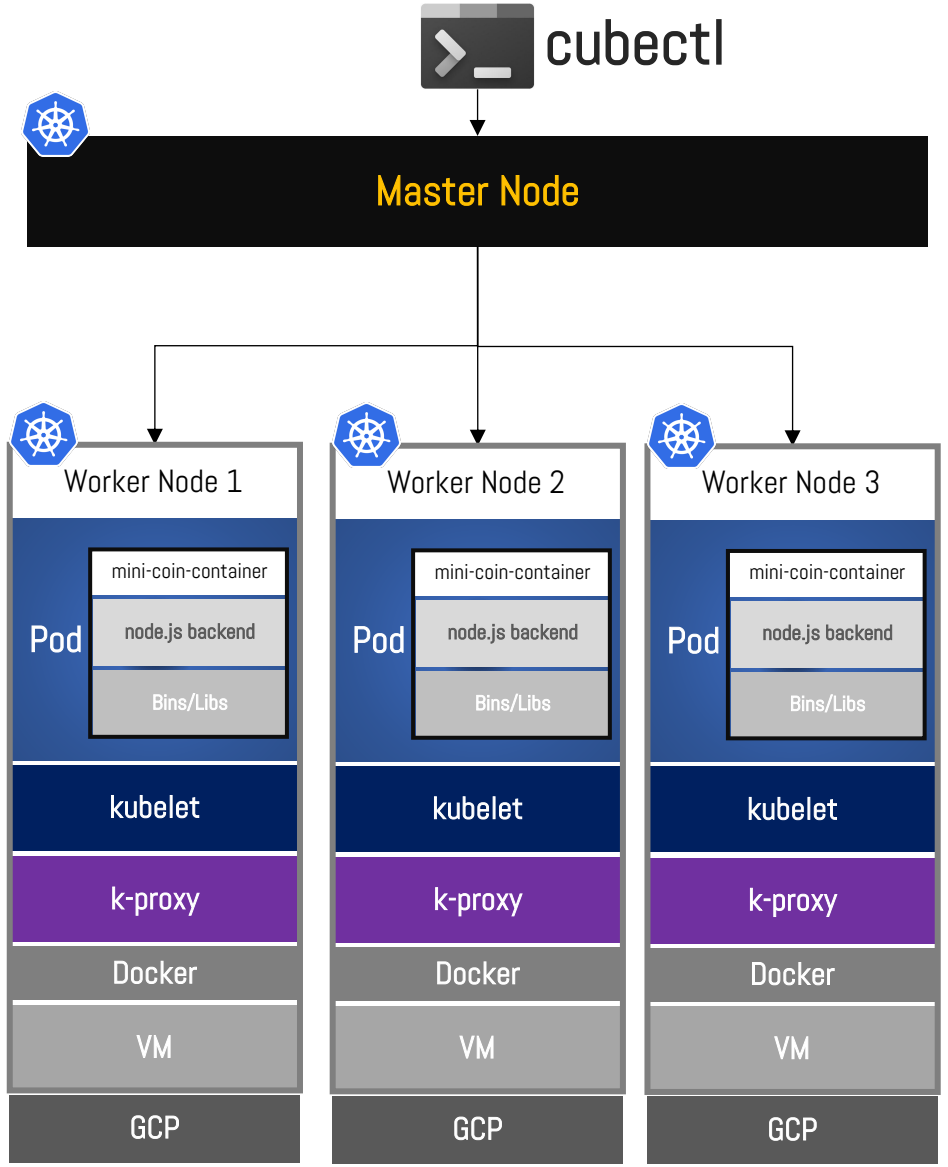
- Pods are the smallest, most basic deployable objects in Kubernetes
  - Container < Pod < VM
- A Pod represents a single instance of a running process in your cluster
- Pods contain one or more Docker containers.
  - Usually one as in this example!
- Pods are created by the Kubernetes scheduler



Kubernetes deployments

- A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold containerised apps.

```
Deployment plan
replicas: 3
image: mini-coin-image
port: 3000
```



Master: Controller manager

- In Kubernetes, a controller watches the shared state of the cluster through the API server and makes changes attempting to move the current state towards the desired state
- Some types of these controllers are:
  - Node controller: Responsible for noticing and responding when nodes go down.
  - Job controller: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
  - Endpoints controller: Populates the Endpoints object (that is, joins Services & Pods).
  - Service Account & Token controllers: Create default accounts and API access tokens for new namespaces.

The cluster store: etcd

- Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
- Check it out: etcd

- etcd prefers consistency over availability:
  - It will not tolerate a split-brain situation and will halt updates to the cluster in order to maintain consistency
  - Cluster continuous to operate just won't be able to update

The scheduler

- It watches the API server for new work tasks and assign to appropriate nodes.
- Nodes (workers) are ranked according to criteria (ranking system is complex)
- Are you looking for a project idea?
  - Good area for research

Workers

- The workforce of the cluster
- Tasks:
  - Watch the API for new work assignments
  - Execute new work assignments
  - Report back to the control plane (via API)

Packaging apps for Kubernetes

- To run on Kubernetes:
  1. Package as a container
  2. Wrapped in a Pod
  3. Deployed via a declarative manifest (YAML)

Scaling pods

- Pods are the minimum unit of scheduling
- If you need to scale you add pods
  - You do not scale by adding more containers in the same pod
  - Pod sharing is for cases that container share resources

Inspecting a deployment manifest file

- **Replicas:**
  - The number of instances of the pod that the deployment should keep alive.
- **template:**
  - When the deployment creates pods, it will create them using this template (image and containerPort)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mini-hi-deployment
  labels:
    app: minihi
spec:
  replicas: 3
  selector:
    matchLabels:
      app: minihi
  template:
    metadata:
      labels:
        app: minihi
    spec:
      containers:
      - name: minihi
        image: steliosot/mini-hi:1
        imagePullPolicy: Always
        ports:
        - containerPort: 3000
```

Inspecting a service manifest file

- **kind**: Service (load balancer)
- **spec**:
  - Mapping container port with pod
    - Pod 80 ⟶ Container: 3000
  - Selector defines the app (from the deployments file)

```
apiVersion: v1
kind: Service
metadata:
  name: mini-hi-service
  labels:
    app: mini-hi-service
spec:
  type: LoadBalancer
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app: minihi
  sessionAffinity: None
```

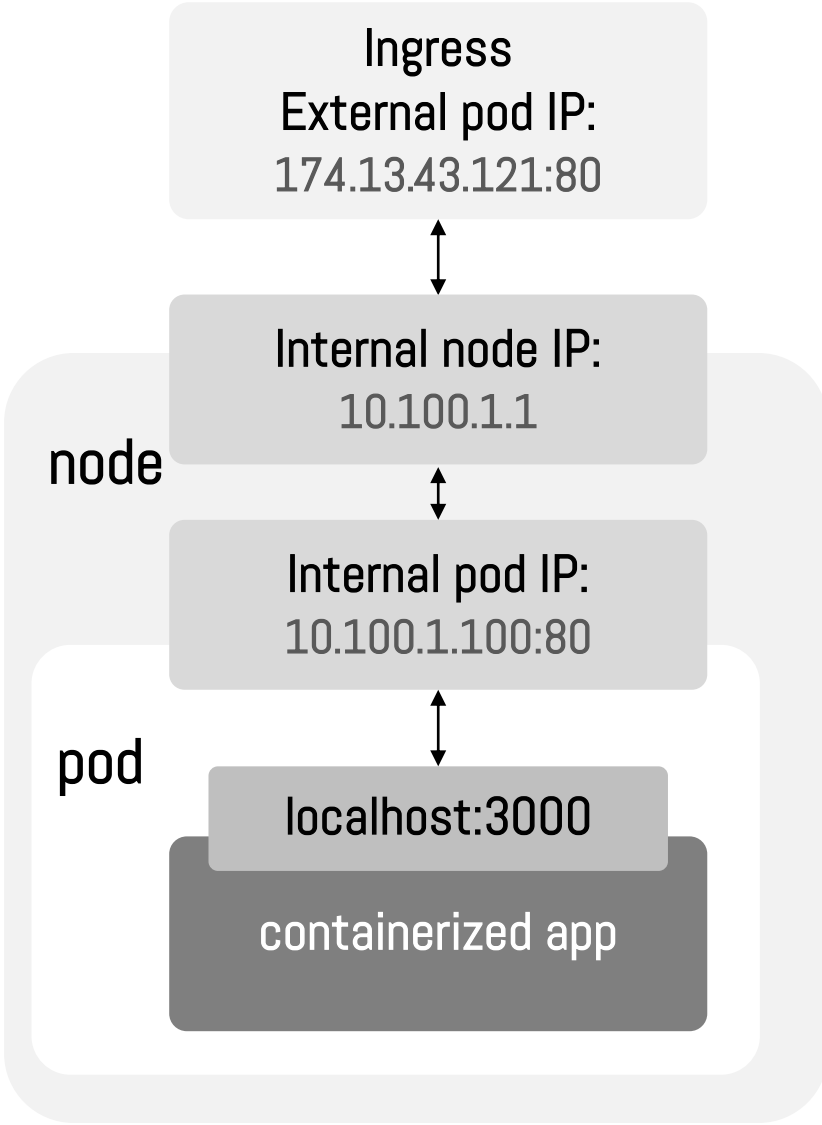
Bonus Lab: Google Kubernetes Engine

Google Kubernetes Engine (GKE)

- Kubernetes is an open-source software for container orchestration
- Focused on automated:
  - Scheduling
  - Scaling
  - Self-healing

GKE Key features

- Pods:
  - A Pod represents a single instance of a running process in your cluster.
- An Ingress controller;
  - A **specialized load balancer** for Kubernetes
  - It abstracts away the complexity of Kubernetes application traffic routing and provides a bridge between Kubernetes services.

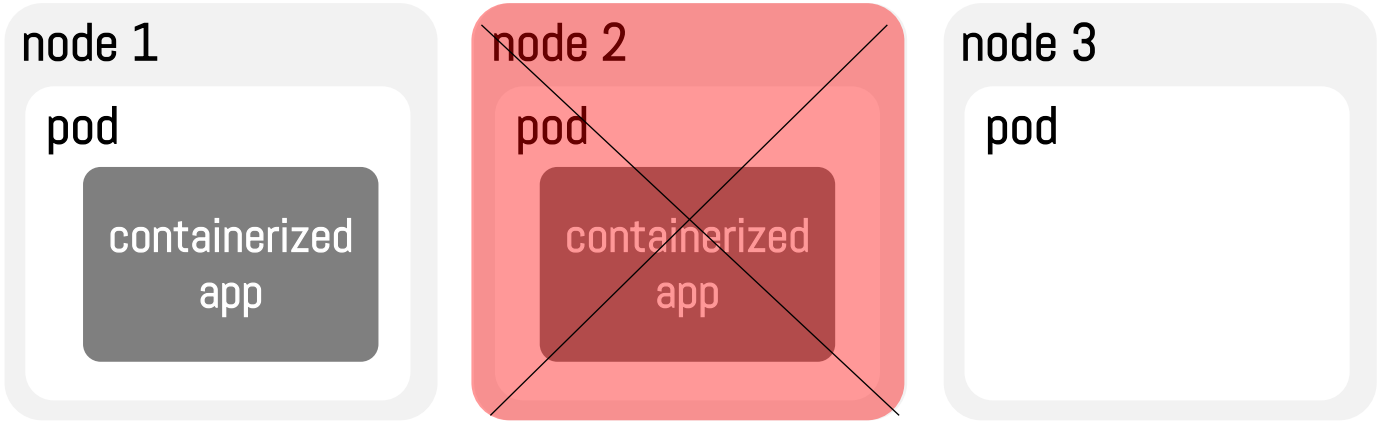


GKE Deployments

- A Kubernetes Deployment is a file used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application.
  - `app:mini-hi`
  - `replicas:1`

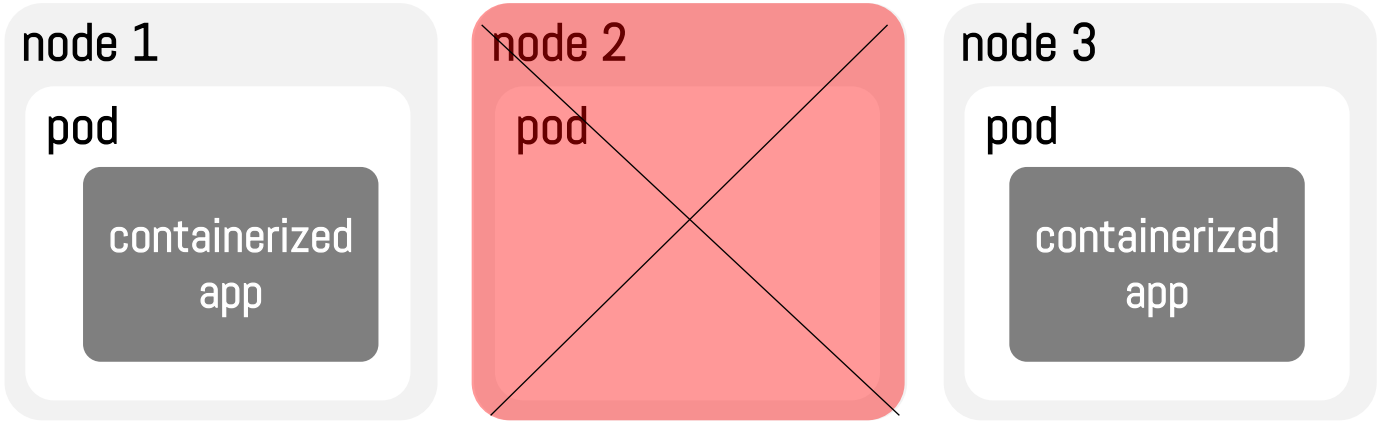
GKE Deployments

- A Kubernetes Deployment is a file used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application.
  - `app:mini-hi`
  - `replicas:2`
  - Failure?



GKE Deployments

- A Kubernetes Deployment is a file used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application.
  - `app:mini-hi`
  - `replicas:2`

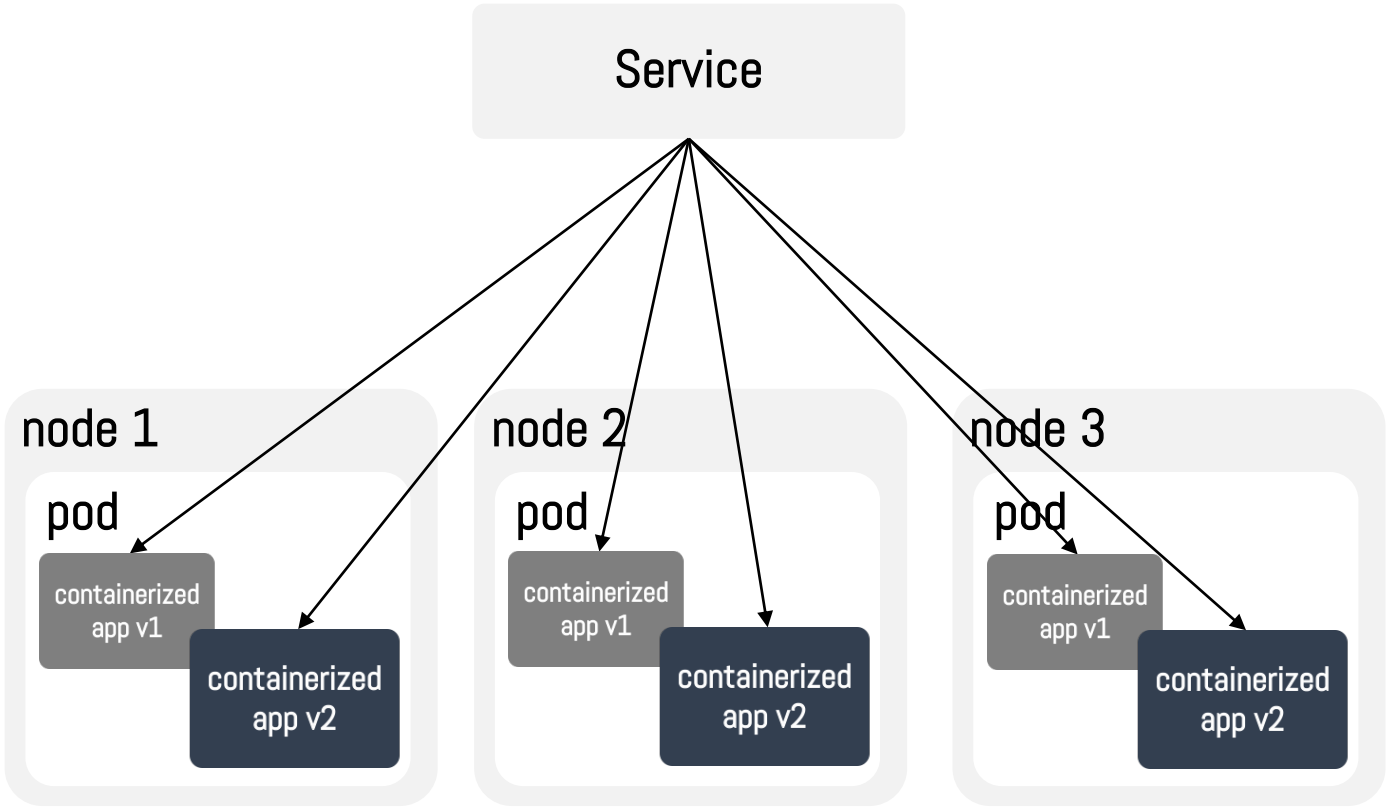


GKE Deployments

- A Kubernetes Deployment is a file used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application.
  - `app:mini-hi`
  - `replicas:10`

Rolling updates

- You can update your software (containers) on the fly.
  - From
    - `image: mini-hi:1`
  - To
    - `image: mini-hi:2`



Todays lab

When you finished, delete your Kubernetes cluster!

Bonus Lab

- Class-Bonus

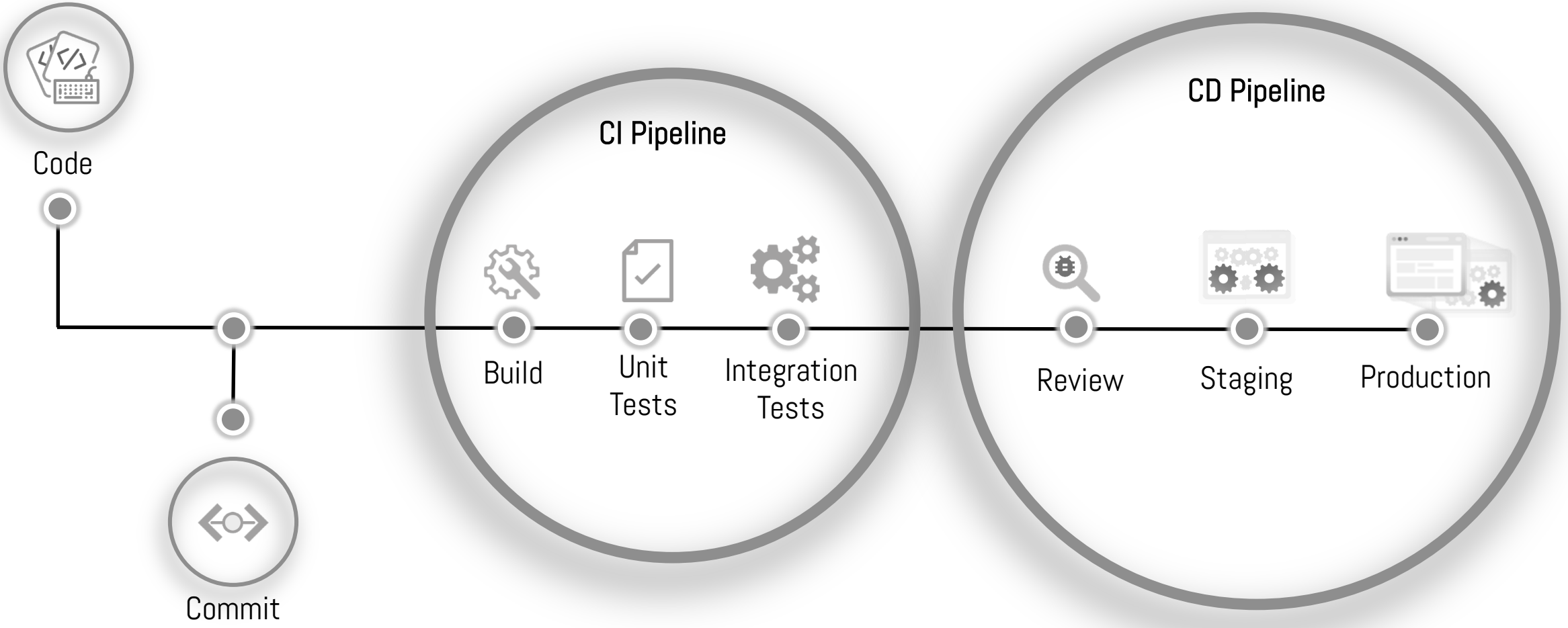
- Did you complete Bonus class?
  - Deploy your MiniPost using a private DockerHub image

Take home

- What is Kubernetes?
- Why it is important to use Kubernetes?
- What problems does it solve?
- How do you setup a GKE cluster and deploy an app?

What is Infrastructure as Code?

DevOps: from your laptop to production



What is infrastructure as code (IaC)?

- Infrastructure as Code (IaC) is the managing and provisioning of infrastructure **through code instead of manual processes**
- We create configuration files that contain infrastructure specifications, which makes it easier to edit and distribute configurations
  - It also ensures that you provision the same environment every time
- By codifying and documenting your configuration specifications, IaC aids configuration management and helps you to avoid undocumented, ad-hoc configuration changes

What is infrastructure as code (IaC)?

- Version control is an important part of IaC, and your configuration files should be under source control
- Deploying your infrastructure as code also means that you can divide your infrastructure into **modular components** that can then be combined in different ways through automation

IaC deployment

- Automating infrastructure provisioning with IaC means that developers don't need to manually provision and manage servers, operating systems, storage, and other infrastructure components each time they develop or deploy an application.
  - You use your code to deploy using configuration files

Conf files: Declarative vs. imperative

- Declarative approach:
  - Defines the desired state of the system, including what resources you need and any properties they should have, and an IaC tool will configure it for you.

```
S3Bucket:
   Type: AWS::S3::Bucket
   Properties:
     BucketName: mybucketname
```

- Imperative approach:
  - Defines the specific commands needed to achieve the desired configuration, and those commands then need to be executed in the correct order.

```
$ aws s3api create-bucket --bucket "mybucketname"
```

Declarative definitions: The preferred approach

- A definition file specifies the what an environment requires and not necessarily the how
  - It may define the specific version and configuration of a server component as a requirement but does not specify the process for installing and configuring it
  - This abstraction allows for greater flexibility and helps to reduce the technical debt of maintaining imperative code, such as deployment scripts, that can accrue over time
- Different platforms support different, and often multiple, file formats, such as YAML, JSON, and XML

Benefits of CI/CD using IaC

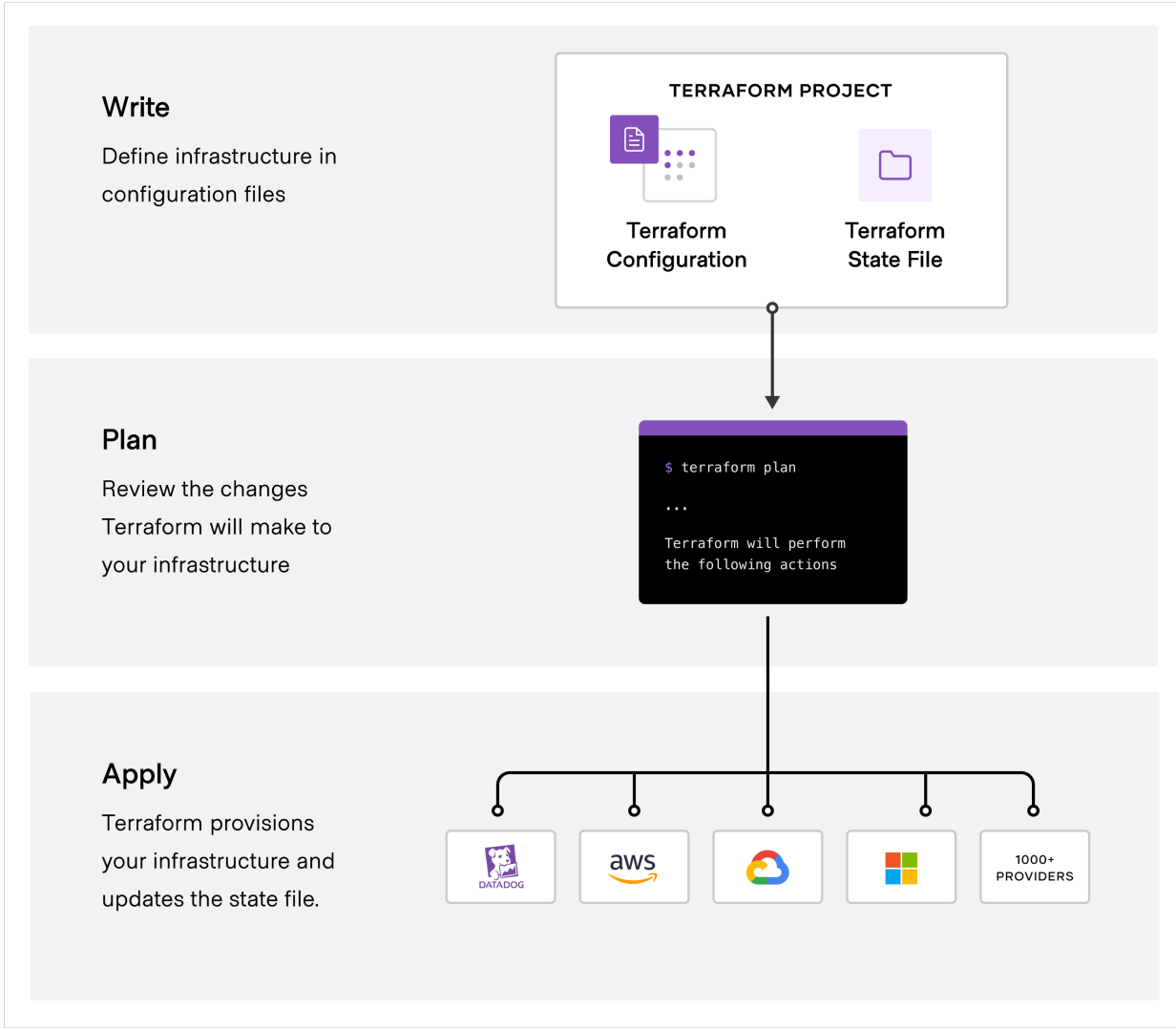
- Cost reduction:
  - Manage your infrastructure better
- Increase in speed of deployments
  - Deploy fast and efficient
- Reduce errors
  - Manage IaC configuration files to allow reproducible environments
- Improve infrastructure consistency
  - Declarative definitions help you to redeploy again and again
- Eliminate configuration drift
  - an infrastructure become increasingly different over time, usually due to manual changes and updates

IaC → DevOps → CI/CD

- CI/CD relies on ongoing automation and continuous monitoring throughout the application lifecycle, from integration and testing to delivery and deployment
- IaC is an important part of implementing DevOps practices and CI/CD.
  - IaC takes away most of the provisioning work from developers, who can execute a script to have their infrastructure ready to go
  - That way, application deployments aren't held up waiting for the infrastructure, and the sysadmins aren't managing time-consuming manual processes

What is Terraform?

- Terraform is an infrastructure as code tool
- You define both cloud and on-premises resources in human-readable configuration files that you can version, reuse, and share
- You can then use a consistent workflow to provision and manage your infrastructure throughout its lifecycle
- Terraform can manage every small detail of a deployment plan



Terraform tool

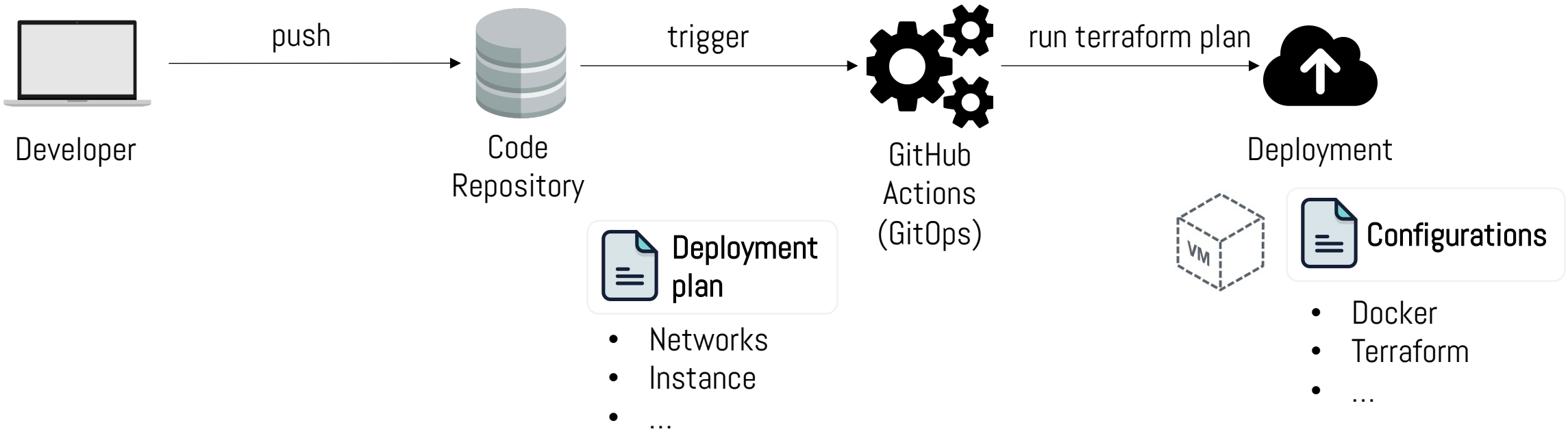
- Terraform community have already written **more than 1700 providers** to manage thousands of different types of resources and services, and this number continues to grow.
  - You can find all publicly available providers on the Terraform Registry, including Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, and many more.
- The core Terraform workflow consists of three stages:
  - **Write:** You define resources, which may be across multiple cloud providers and services.
    - For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.
  - **Plan:** Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
  - **Apply:** On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies.
    - For example, if you update the properties of a VPC and change the number of virtual machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.

What is GitOps?

- GitOps is an operational framework that takes DevOps best practices used for application development such as version control, collaboration, compliance, and CI/CD tooling, and applies them to infrastructure automation.
- The Terraform integration features in GitLab enable your GitOps / Infrastructure-as-Code (IaC) workflows to tie into GitLab authentication and authorization.

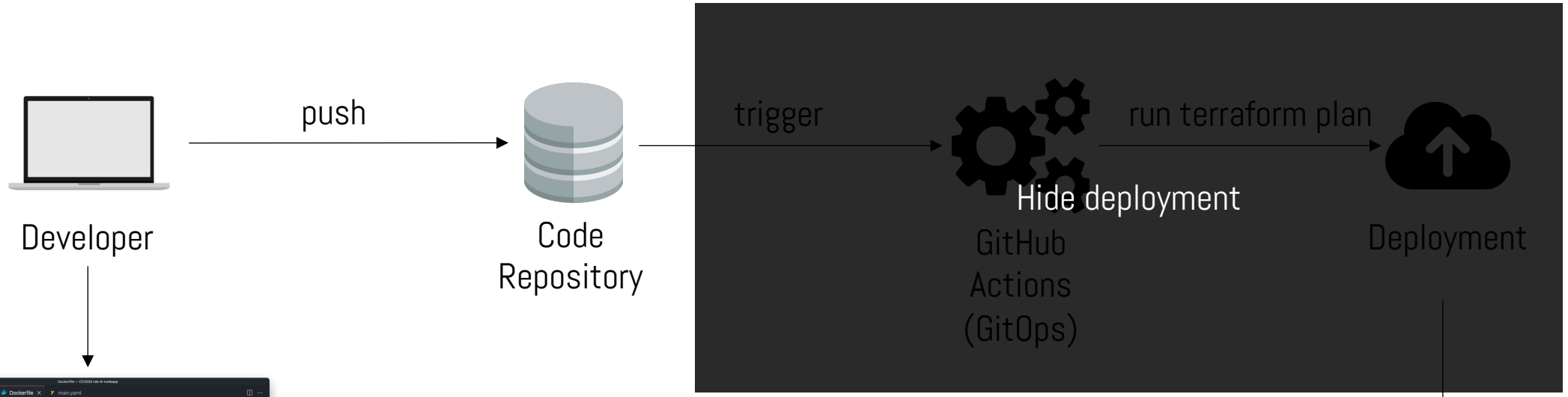
CI/CD pipeline with GitHub actions

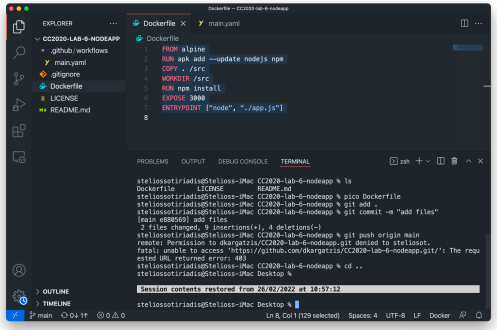
- Setting up an environment (VM)

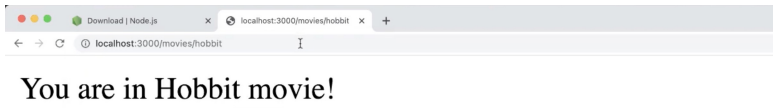


CI/CD pipeline with GitHub actions

- Environment is up! Let's setup a node deployment plan







Demo: Building a GKE cluster

Building a GKE cluster

- Intro to Kubernetes
- Create a cluster and deploy containers
- Building Docker images and pushing code to GitHub, DockerHub and Kubernetes

Practical Session

Deployment using Docker

- Run the tutorials:
  - Introduction to GKE
  - GKE Command Line tool
  - Using GitHub, DockerHub and Docker to deploy and run containers
- Exercises:
  - Deploy your Docker containers in GKE