Kubernetes Workshop

Oct 28, 2024
Ibergrid - Porto

Mastering the Essentials

Kubernetes Fundamentals

Miguel Viana <mviana@lip.pt>

Why Kubernetes?

amadeus

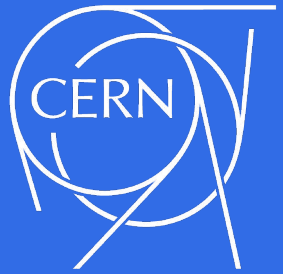
“

Amadeus, provides IT solutions to the travel industry around the world, found itself in need of a new platform for the **5,000 services supported** by its service-oriented architecture. [...] Among the company's goals: to **increase automation in managing its infrastructure**, **optimize the distribution of workloads**, **use resources more efficiently**, and **adopt new technologies more easily**.

”

Scalability | Workload Distribution | Resource Efficiency | Automation

https://kubernetes.io/case-studies/amadeus



“CERN experiences extreme peaks in its workloads during periods prior to big conferences, and **needs its infrastructure to scale to those peaks**. "We want to have a more hybrid infrastructure, where we have our on premise infrastructure but can make use of public clouds temporarily when these peaks come up,". "**We've been looking to new technologies that can help improve our efficiency in our infrastructure** so that we can dedicate more of our resources to the actual processing of the data."”

Scalability **Hybrid Cloud Support** **Resource Efficiency**

https://kubernetes.io/case-studies/cern

“

As an artificial intelligence research lab, OpenAI needed infrastructure for deep learning that would allow experiments to be run either in the cloud or in its own data center, and to **easily scale**. **Portability**, **speed**, and **cost** were the main drivers.

”

Portability **Scalability** **Speed**

https://kubernetes.io/case-studies/openai

Booking.com

"In 2016, Booking.com migrated to an OpenShift platform, which gave product developers faster access to infrastructure. But because Kubernetes was abstracted away from the developers, the infrastructure team became a "knowledge bottleneck" when challenges arose. **Trying to scale that support wasn't sustainable.**

After a year operating OpenShift, the platform team decided to build its own vanilla Kubernetes platform—and ask developers to learn some Kubernetes in"

https://kubernetes.io/case-studies/booking-com

Booking.com

"knowledge bottleneck" when challenges arose. Trying to scale that support wasn't sustainable.

After a year operating OpenShift, the platform team decided to build its own vanilla Kubernetes platform—and ask developers to learn some Kubernetes in order to use it. **Developers need to do some learning**, and we're going to do everything we can to make sure they have access to that knowledge.

Despite the learning curve, there's been a great uptick in adoption of the new Kubernetes platform. Before containers, creating a new service could take a

https://kubernetes.io/case-studies/booking-com

Booking.com

order to use it. Developers need to do some learning, and we're going to do everything we can to make sure they have access to that knowledge.

Despite the learning curve, there's been a great uptick in adoption of the new Kubernetes platform. Before containers, creating a new service could take a couple of days if the developers understood Puppet, or weeks if they didn't. **On the new platform, it can take as few as 10 minutes**. About 500 new services were built on the platform in the first 8 months.

https://kubernetes.io/case-studies/booking-com

Let's get started!

How to interact with a kubernetes cluster?

Connect to cluster bastion

```
ssh <INSTANCE>.ibergrid.gcloud.a.incd.pt \
    -l <YOUR_USER>
```

How to interact with a kubernetes cluster?

Get familiar with kubectl command

```
$ kubectl  # or simply 'k'
```

How to interact with a kubernetes cluster?

Get familiar with kubectl command

kubectl is the primary command-line tool for interacting with Kubernetes clusters.

You can use it to perform various tasks, such as:

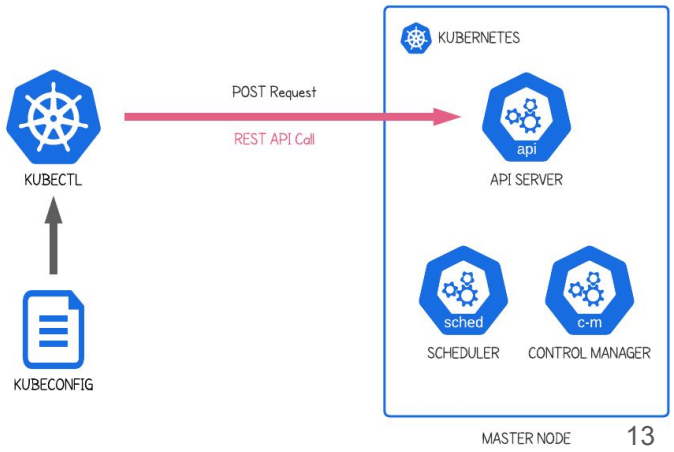
- List and manage nodes
- Deploy and manage applications
- View logs and troubleshoot issues

Beyond kubectl: Other deployment techniques in K8s

YAML/JSON

Standard format for defining resources (manifests).

kubectl apply -f



Beyond `kubectl`: Other deployment techniques in K8s

Helm Charts

Packages application code, configs, and deployment manifests (think app store for Kubernetes).

```
helm                    install                    [...]
```

Beyond `kubectl`: Other deployment techniques in K8s

`Kustomize`

Customize deployments for different environments (dev/test/prod) with overlays.

Namespaces
Organizing Your K8s Cluster

Namespaces: Organizing your K8s cluster

What is a K8s namespace?

Namespaces may be seen as virtual clusters within a physical Kubernetes cluster. They provide a way to **logically isolate resources** like pods, deployments, and services.

Namespaces: Organizing your K8s cluster

K8s namespaces benefits

Grouped Applications: Organize related applications together.

Namespaces: Organizing your K8s cluster

K8s namespaces benefits

Multi-Tenancy: Enable multiple teams or users to share the same physical cluster securely. **Each namespace acts as a separate environment** with its own resources and access control.

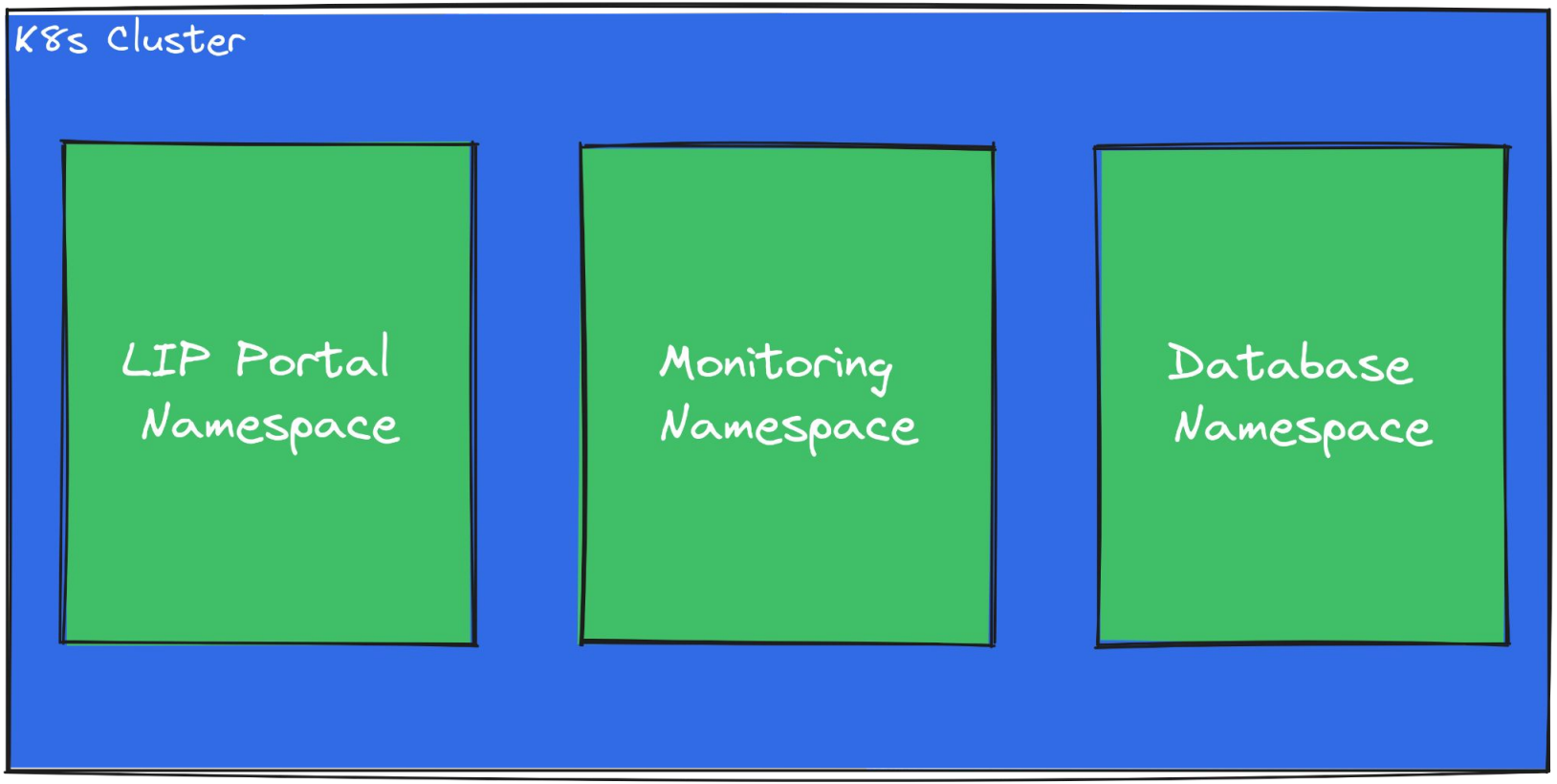
Namespaces: Organizing your K8s cluster

`K8s namespaces benefits`

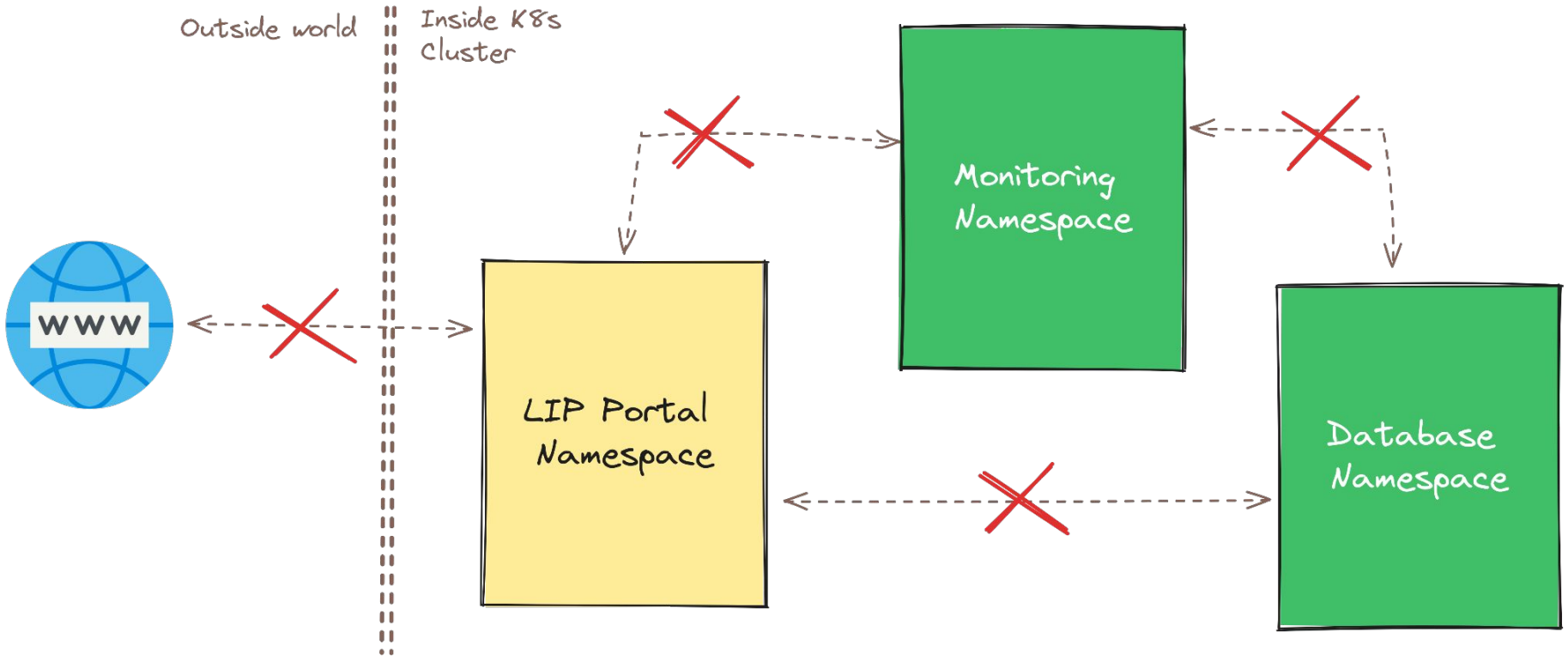
Resource Quotas and Access Control: Define resource limits (CPU, memory) and access permissions for each namespace. **This ensures efficient resource utilization and prevents unauthorized access.**

Let's deploy our application!

Our deployment: Current state



Our deployment: Current state



Lab 0

Exploring Your Cluster with Kubectl

Exploring Your K8s Cluster with kubectl

List cluster nodes

```
$ kubectl get nodes
```

```
NAME             STATUS   ROLES           AGE   VERSION
minikube-bob1    Ready    control-plane   20h   v1.28.3
```

Exploring Your K8s Cluster with kubectl

Describe a node

```
$ kubectl describe node minikube-bob1
```

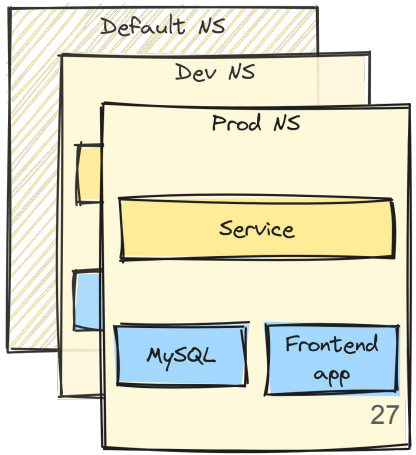
```
.Name:              minikube-bob1
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=minikube-bob1
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/cri-dockerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                  [...]
```

Exploring Your K8s Cluster with kubectl

List namespaces

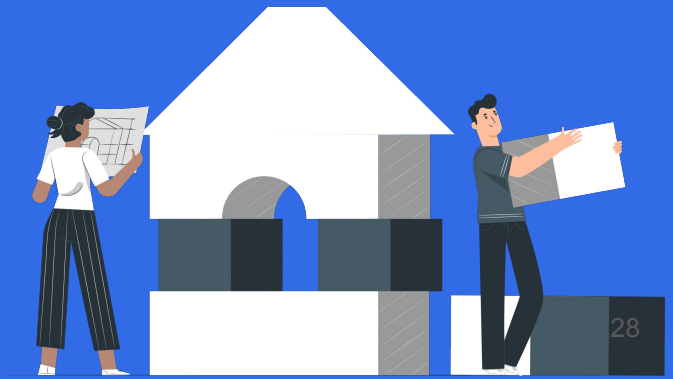
```
$ kubectl get namespaces
```

```
NAME              STATUS   AGE
default           Active   20h
ingress-nginx     Active   20h
kube-node-lease   Active   20h
kube-public       Active   20h
kube-system       Active   20h
```



Lab 1

Deploying our first application



Deploying our first application

Create a new Pod to run Nginx

```
$ kubectl run my-nginx-pod \
    --image=nginx:latest \
    --port=80
```

```
pod/my-nginx-pod created
```

Deploying our first application

Listing Pods

```
$ kubectl get pods
```

```
NAME            READY   STATUS    RESTARTS   AGE
my-nginx-pod    1/1     Running   0          112s
```

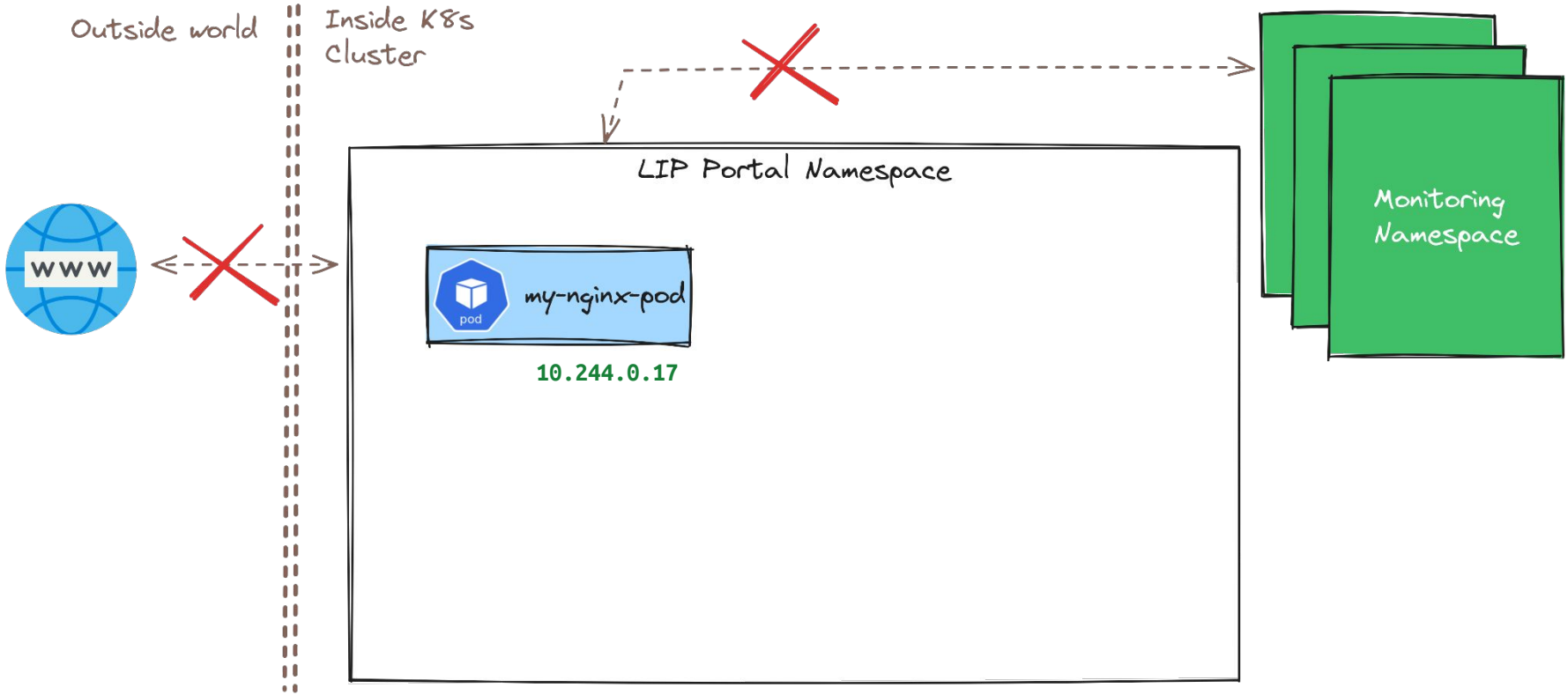
Deploying our first application

Listing Pods in detail

```
$ kubectl get pods -o wide
```

```
NAME              READY     STATUS     RESTARTS     AGE        IP             NODE
my-nginx-pod      1/1       Running    0            4m31s      10.244.0.17    minikube-bob1
```

Our deployment: Current state



Lab 2

Exposing the application

Exposing the application

Expose the pod as a Service

```
$ kubectl expose pod my-nginx-pod \
    --port=80 \
    --name=nginx-svc
```

```
service/nginx-svc exposed
```

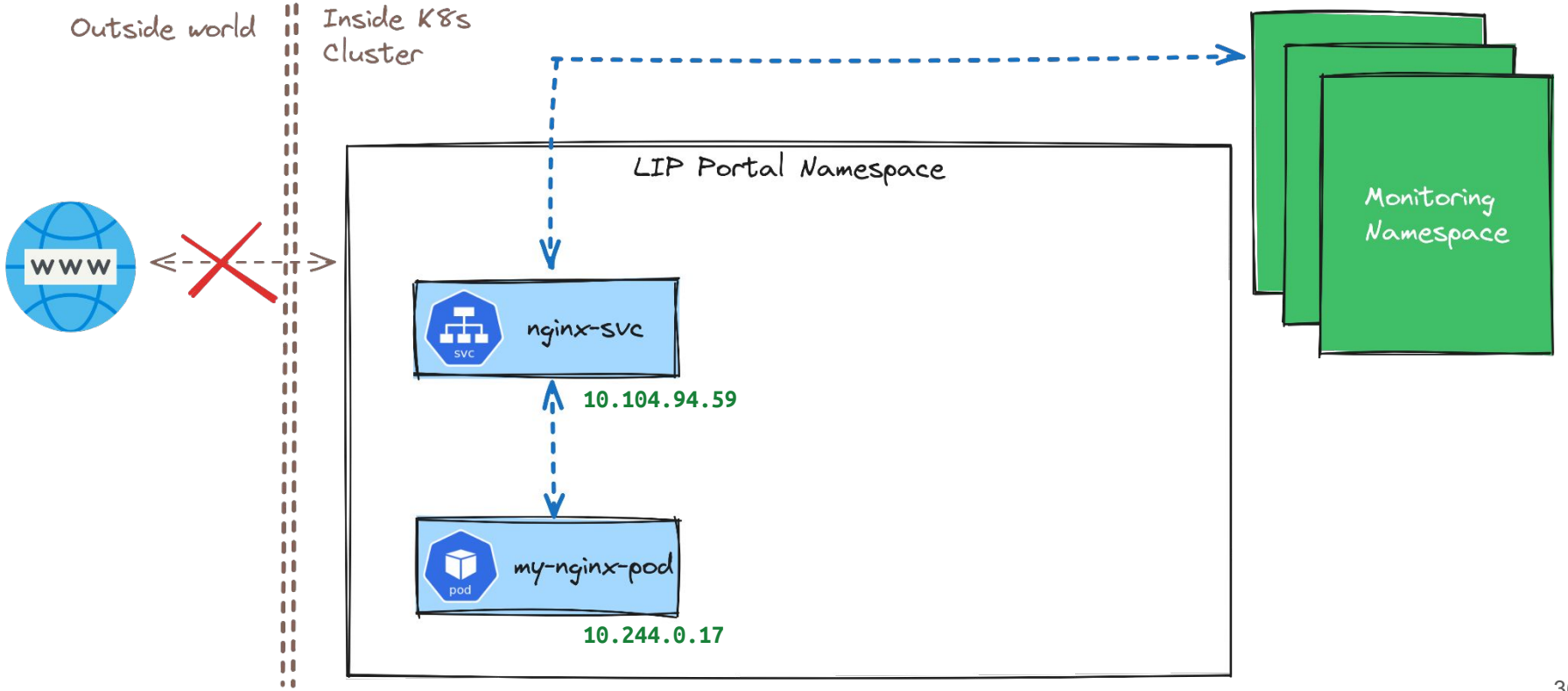
Exposing the application

Verify your service is created

```
$ kubectl get services
```

```
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
nginx-svc    ClusterIP   10.104.94.59   <none>        80/TCP    7m4s
```

Our deployment: Current state



Lab 3

Advanced Exposure with Ingress

Advanced Exposure with Ingress

Expose a service to the world

```
$ kubectl create ingress nginx-ing \
    --rule="example.com/*=nginx-svc:80"
```

```
ingress.networking.k8s.io/nginx-ing created
```

Advanced Exposure with Ingress

Explain the command

```
kubectl create ingress nginx-ing \
  --rule="example.com/*=nginx-svc:80"
```

`kubectl create ingress`: Command to create a new ingress.

`nginx-ing`: Ingress name.

Advanced Exposure with Ingress

Explain the command

```
kubectl create ingress nginx-ing \
 --rule="example.com/*=nginx-svc:80"
```

`--rule`: Specifying a rule for the Ingress.

`example.com`: Hostname that will trigger this rule.

`*`: This wildcard indicates that the rule applies to any path within the URL. E.g., **`/`**, **`/about`**, **`/products`**

Advanced Exposure with Ingress

Explain the command

```
kubectl create ingress nginx-ing \
  --rule="example.com/*=nginx-svc:80"
```

`nginx-svc:80`: This defines the backend service to which traffic will be directed.

`nginx-svc` Name of the Service that exposes your app pods.

`:80` Port on the Service that handles incoming traffic.

Advanced Exposure with Ingress

Generate a YAML file

```
$ kubectl create ingress nginx-ing \
    --rule="example.com/*=nginx-svc:80" \
    --dry-run=client -o yaml > ingress.yaml
```

Advanced Exposure with Ingress

Generated YAML file

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ing
spec:
  rules:
  - host: example.com
    http:
      paths:
      - backend:
          service:
            name: nginx-svc
            port:
              number: 80
        path: /
        pathType: Prefix
```

Hostname that will trigger this rule

Service that exposes your app pods

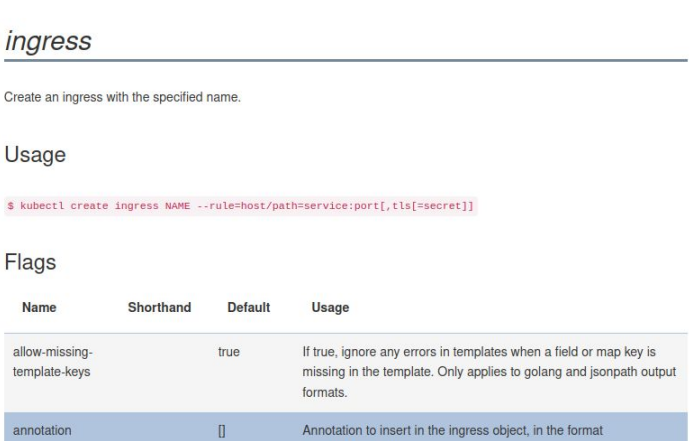
Port that handles incoming traffic

Indicates that the rule applies to any path within

Advanced Exposure with Ingress

Let's check the K8s Documentation

You can refer to the Kubernetes documentation for imperative commands (focusing on the kubectl create ingress section) at: https://kubernetes.io/docs/reference/kubectl/generated/kubectl_create/kubectl_create_ingress/



ingress

Create an ingress with the specified name.

Usage

```
$ kubectl create ingress NAME --rule=host/path=service:port[,tls[=secret]]
```

Flags

Name	Shorthand	Default	Usage
allow-missing-template-keys		true	If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.
annotation		[]	Annotation to insert in the ingress object, in the format

Create a single ingress called 'simple' that directs requests to foo.com/bar to svc # svc1:8080 with a tls secret "my-cert"

```
kubectl create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"
```

Create a catch all ingress of "/path" pointing to service svc:port and Ingress Class as "otheringress"

```
kubectl create ingress catch-all --class=otheringress --rule="/path=svc:port"
```

Create an ingress with two annotations: ingress.annotation1 and ingress.annotations2

```
kubectl create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla
```

Create an ingress with the same host and multiple paths

```
kubectl create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
```

Advanced Exposure with Ingress

Expose a service with SSL cert

```
$ kubectl create ingress nginx-ing \
    --rule="example.com/*=nginx-svc:80, tls=nginx-secret" \
    --annotation cert-manager.io/cluster-issuer=<issuer_name>
```

```
ingress.networking.k8s.io/nginx-ing created
```

Advanced Exposure with Ingress

Explain the command

```
kubectl create ingress nginx-ing \
  --rule="example.com/*=nginx-svc:80" \
  --tls=nginx-secret \
  --annotation cert-manager.io/cluster-issuer=<issuer_name>
```

`--tls`: specifies the secret name where the certificate will be stored

`--annotation cert-manager.io/cluster-issuer`: instruct the `cert-manager controller` to request a TLS certificate from a pre-configured certificate `issuer` within the Kubernetes cluster.

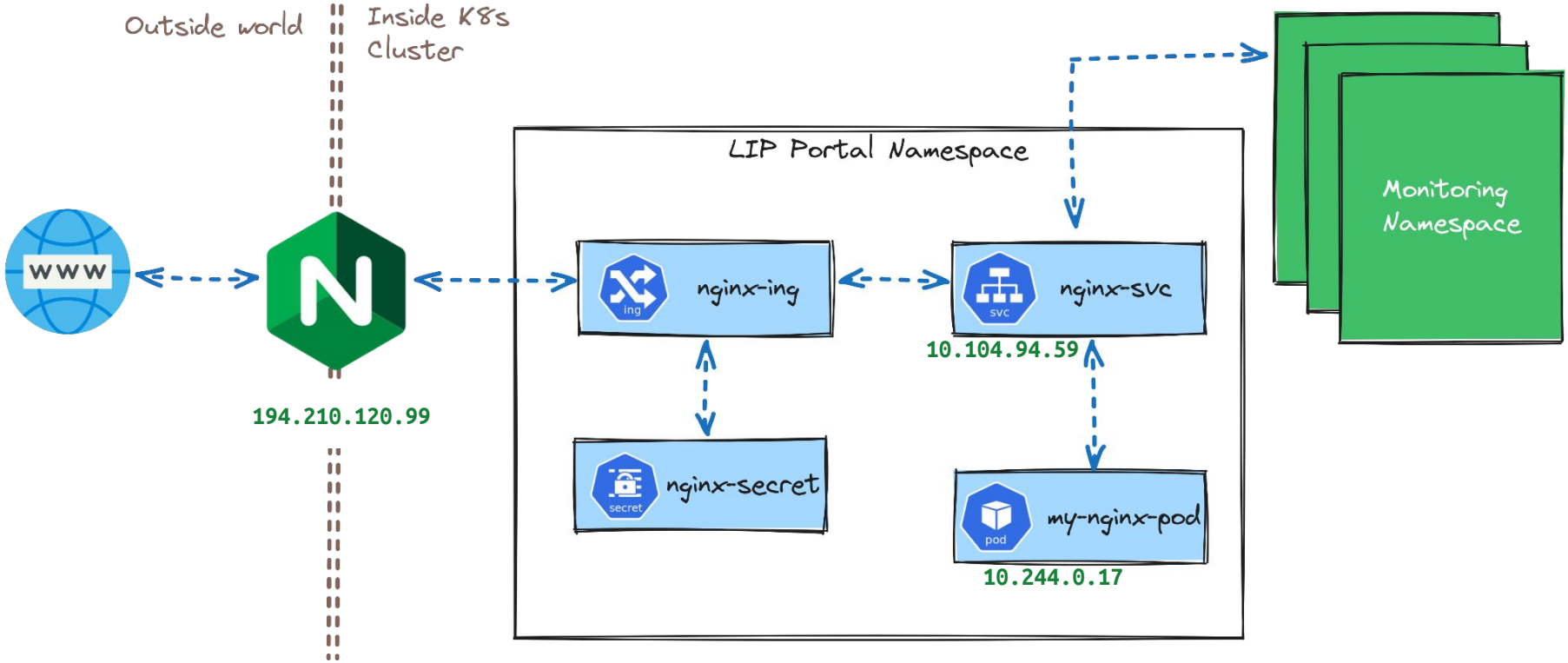
Advanced Exposure with Ingress

List configured certificate issuers

```
$ kubectl get clusterissuers
```

```
NAME              READY   AGE
lip-sectigo       True    12d
lets-encrypt      True    150d
```

Our deployment: Current state



Lab 4

Scaling Up with Deployments

Scaling Up with Deployments

Create a deployment with Nginx app

```
$ kubectl create deployment my-nginx \
    --image=nginx \
    --replicas=2
```

```
deployment.apps/my-nginx created
```

Scaling Up with Deployments

Let's check the deployment resources

```
$ k get deployment,pods
```

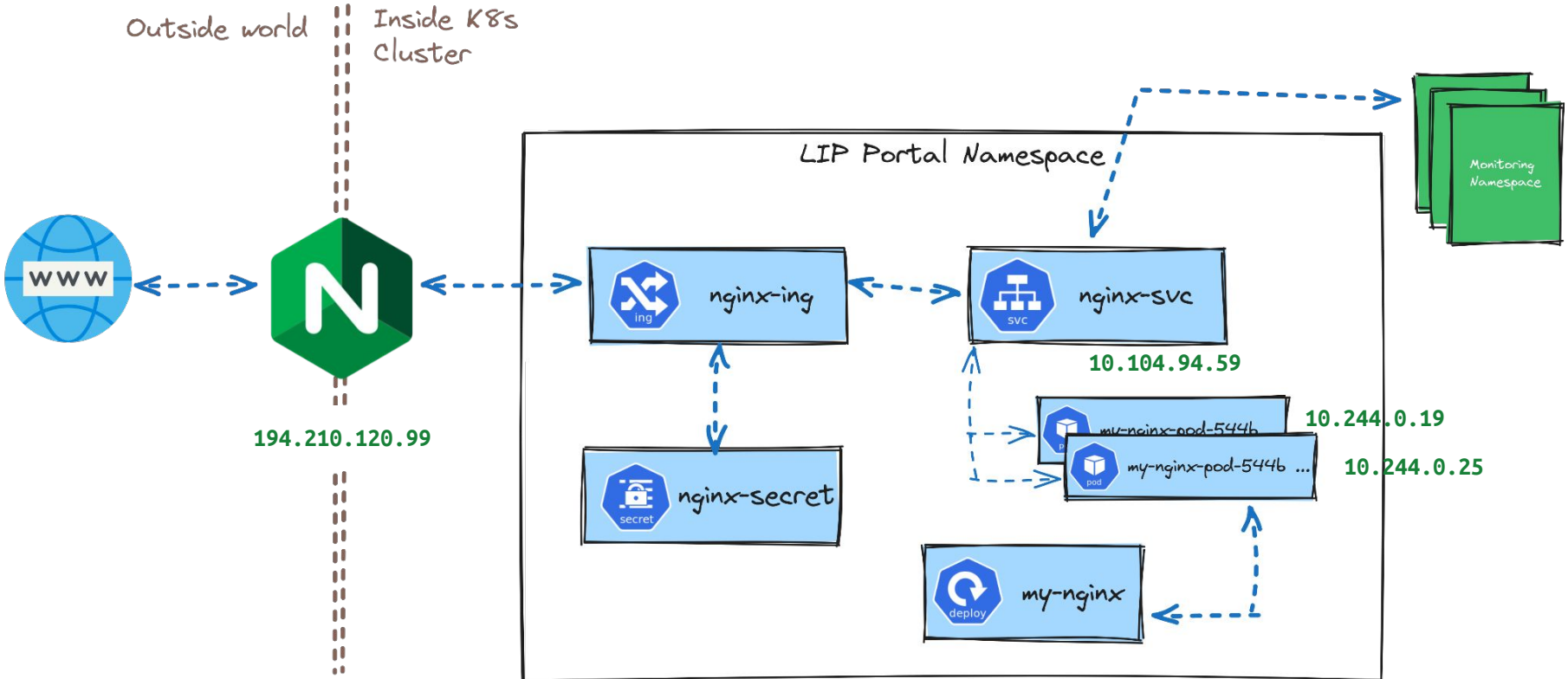
```
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-nginx    2/2     2            2           90s

NAME                                READY   STATUS    RESTARTS   AGE
pod/my-nginx-544b86ccd5-4wkkq       1/1     Running   0          90s
pod/my-nginx-544b86ccd5-tzpc4       1/1     Running   0          90s
```

Scaling Up with Deployments

Advantages of Deployments with Replicas

- High Availability
- Scalability
- Rolling Updates

Our deployment: Current state



Lab 5

Managing Configuration and Secrets

Managing Configuration and Secrets

Create a configMap

```
$ kubectl create configmap nginx-config \
  --from-literal=index.html="<html><h1>...</h1></html>"
```

```
configmap/nginx-config created
```

Managing Configuration and Secrets

List all ConfigMaps

```
$ kubectl get configmaps
```

```
NAME                DATA     AGE
kube-root-ca.crt    1        4d23h
nginx-config        1        108s
```

Managing Configuration and Secrets

Check the content of a desired configMap

```
$ kubectl get cm nginx-config -o yaml
```

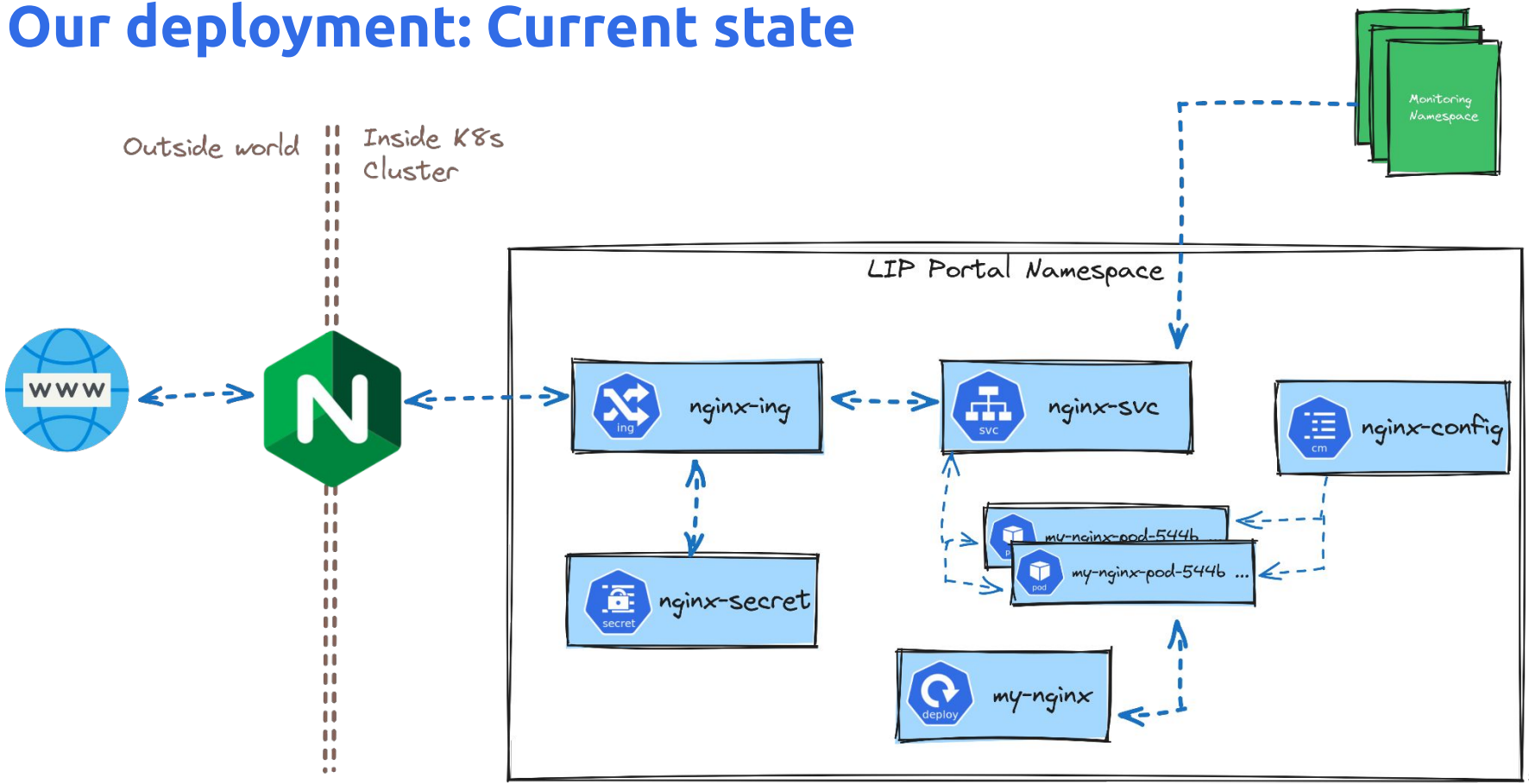
```
apiVersion: v1
data:
  index.html: |
    <html><h1>Hello World </h1></html>
kind: ConfigMap
[...]
```

Managing Configuration and Secrets

Mount the configMap inside Deployment

```
$ kubectl edit deployment my-nginx
```

Our deployment: Current state



Lab 6

Persisting Data with Volumes

Persisting Data with Volumes

Create a Persistent Volume (PV)

> ⚠️ **Warning**
>
> Using **HostPath PVs in production is not recommended** due to tight coupling with the underlying host machine. This is for demonstration purposes only.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-storage
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /path/to/persistent/storage
```

Persisting Data with Volumes

List existing Persistent Volumes (PV)

```
$ kubectl get pv
```

```
NAME            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      AGE
local-storage   1Gi        RWO            Retain           Available   17s
```

Persisting Data with Volumes

Create a Persistent Volume Claim (PVC)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

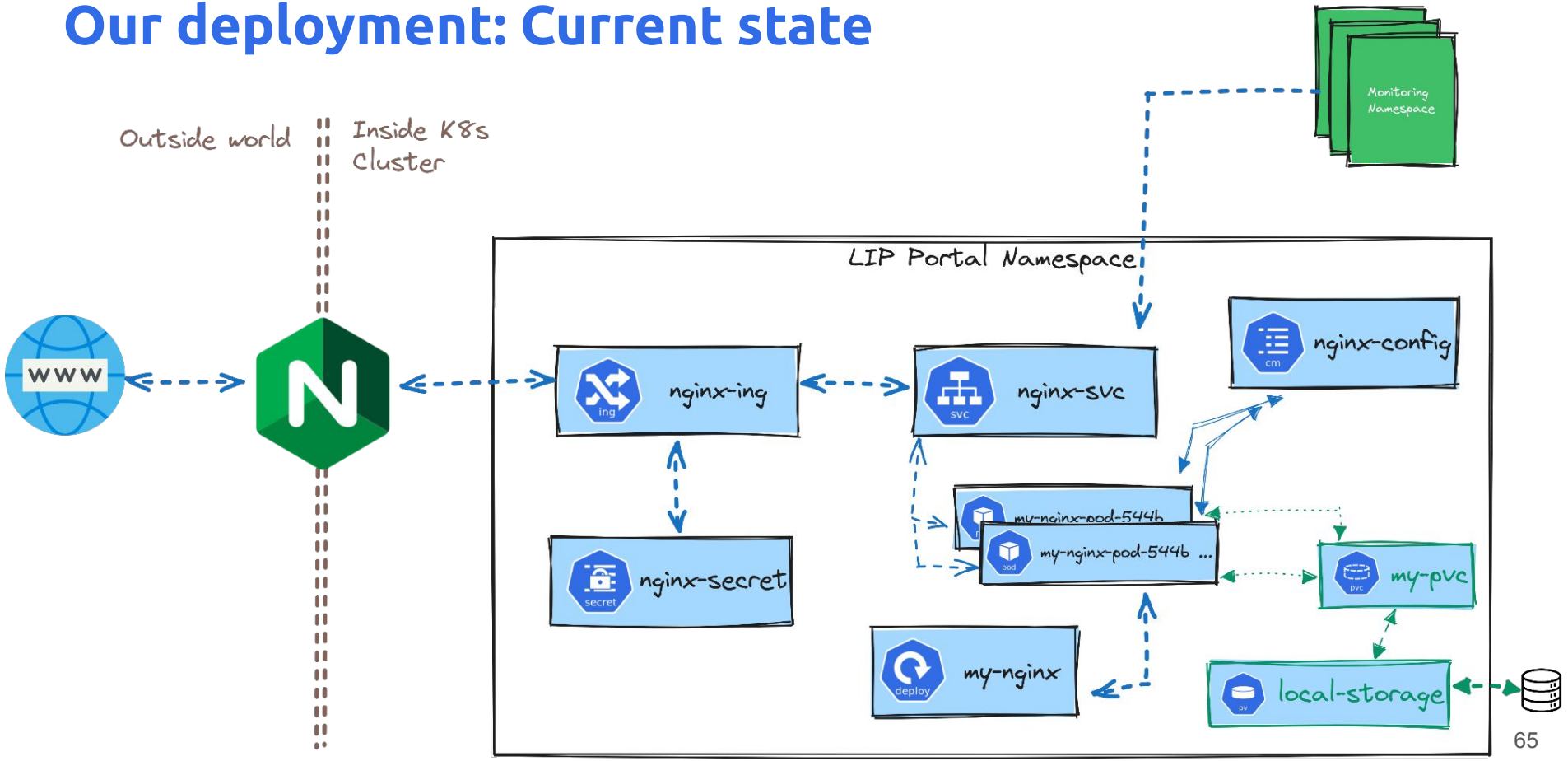
Persisting Data with Volumes

Verify if PVC is created and bounded to the previously created PV

```
$ kubectl get pvc
```

```
NAME      STATUS    VOLUME                    CAPACITY    ACCESS MODES    AGE
my-pvc    Bound     pvc-de071e65-...1af39     1Gi         RWO             2s
```

Our deployment: Current state



Lab 7

Testing a Rollout Deployment

Testing a Rollout Deployment

`kubectl rollout features`

`History`: View the history of revisions for your deployment.

`Pause`: Temporarily halt a rollout in progress.

`Resume`: Resume a paused rollout.

`Status`: Check the current status and progress of a deployment.

`Undo`: Rollback your deployment to a previous revision (if needed).

Testing a Rollout Deployment

Pause a Running Rollout

```
$ kubectl rollout pause deployment my-nginx
```

```
deployment.apps/my-nginx paused
```

Testing a Rollout Deployment

Resume a Paused Rollout

```
$ kubectl rollout resume deployment my-nginx
```

```
deployment.apps/my-nginx resumed
```

Testing a Rollout Deployment

Check Deployment Status

```
$ kubectl rollout status deployment my-nginx
```

```
deployment "my-nginx" successfully rolled out
```

Testing a Rollout Deployment

Check deployment history

```
$ kubectl rollout history deployment my-nginx
```

```
deployment.apps/my-nginx
REVISION  CHANGE-CAUSE
1         <none>
```

Testing a Rollout Deployment

Check Manifests of Specific Revisions

```
$ kubectl rollout history deployment my-nginx \
    --revision=1 \
    -o yaml > revision-<revision number>.yaml
```

```
    apiVersion: apps/v1
    kind: ReplicaSet
    metadata:
      name: my-nginx-544b86ccd5
      namespace: default
    [...]
```

Testing a Rollout Deployment

Undo a Deployment to a Specific Revision

```
$ kubectl rollout undo deployment my-nginx \
    --to-revision=<revision number>
```

Q&A:
Let's Talk about K8s!

Thank you!