Kubernetes Lab & Workshop

2017-07-21 – Martin Danielsson, Haufe-Lexware

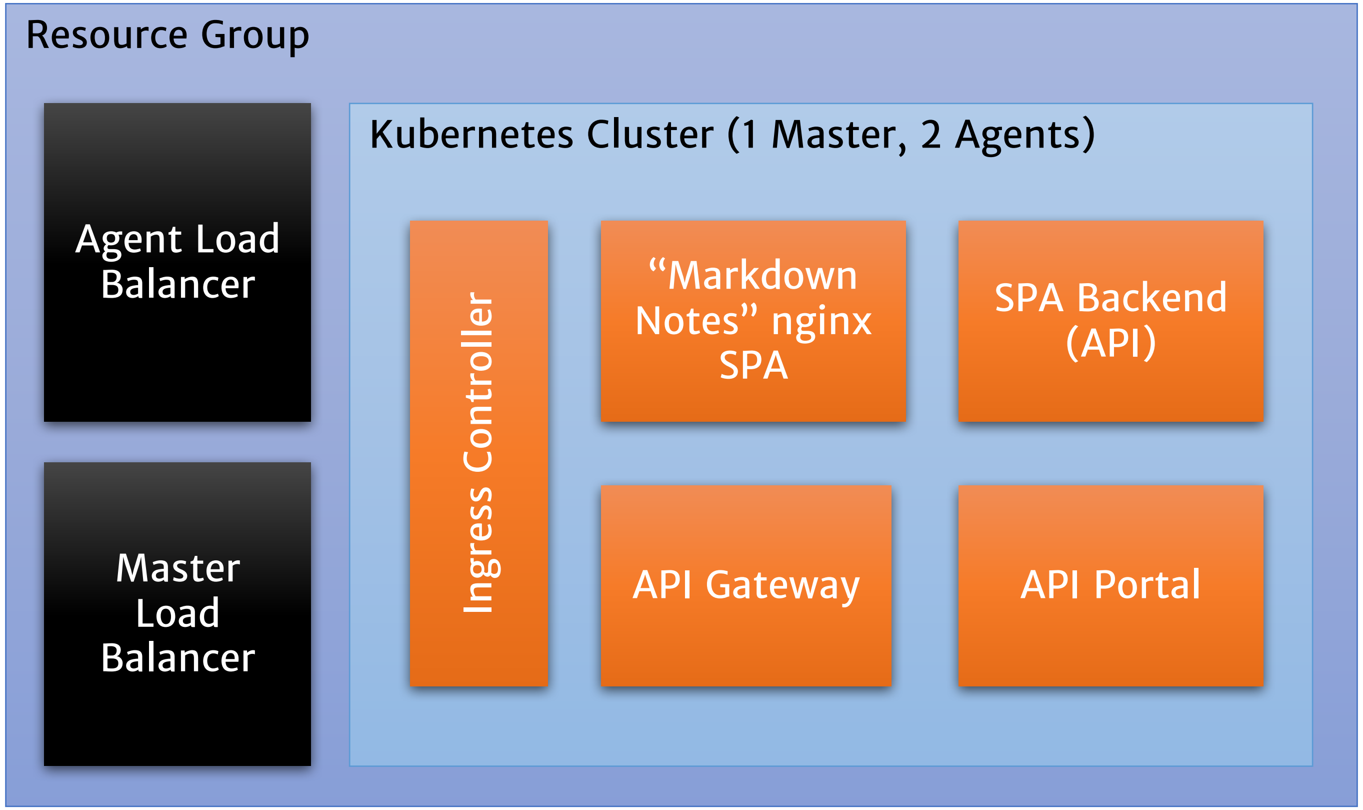
HAUFE.

Objectives for the Workshop

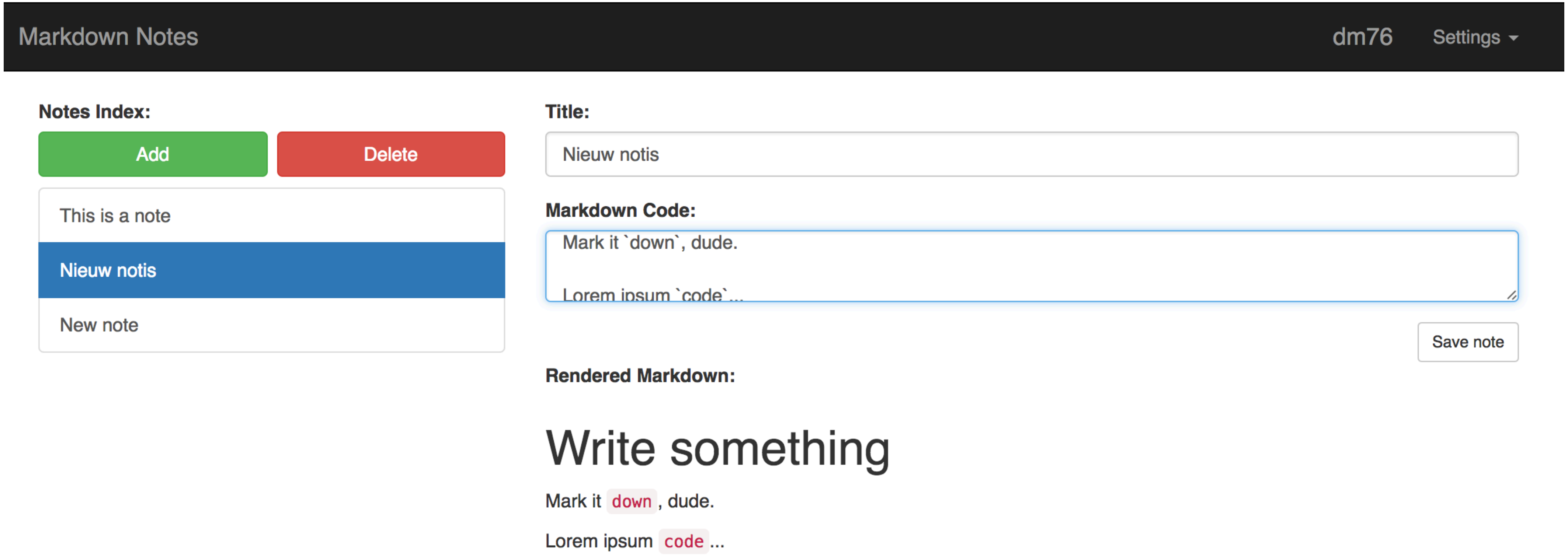
- Get to know Kubernetes concepts – in theory and practice
- Deploy a Kubernetes cluster on Azure Container Services
- Working with a Kubernetes Cluster
- Deploy a multi tier application using real-world techniques
- Few slides, more hands-on

- In: Deployments, Services, Ingress Controllers, ConfigMaps, Secrets
- Out: Namespaces, Storage, RBAC, Helm

What we'll deploy…



Screenshot



This is a note

Nieuw notis

New note

Title: Nieuw notis

Markdown Code:

```
Mark it `down`, dude.

Lorem ipsum `code`...
```

Rendered Markdown:

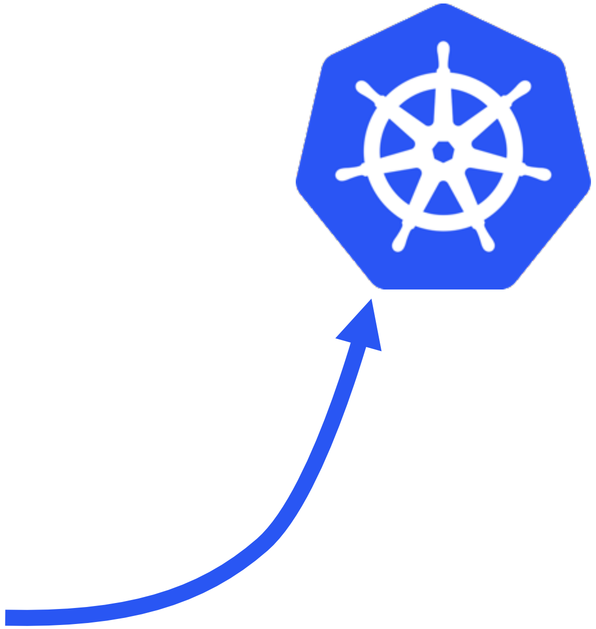
Write something

Mark it down, dude.

Lorem ipsum code ...

Kubernetes Basics

You might have figured,
this is the Kubernetes logo



What is Kubernetes?

"Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure."

http://kubernetes.io/docs/whatisk8s/

Holy smokes!

Which means?

We'll find out today!

We get to run Containers!

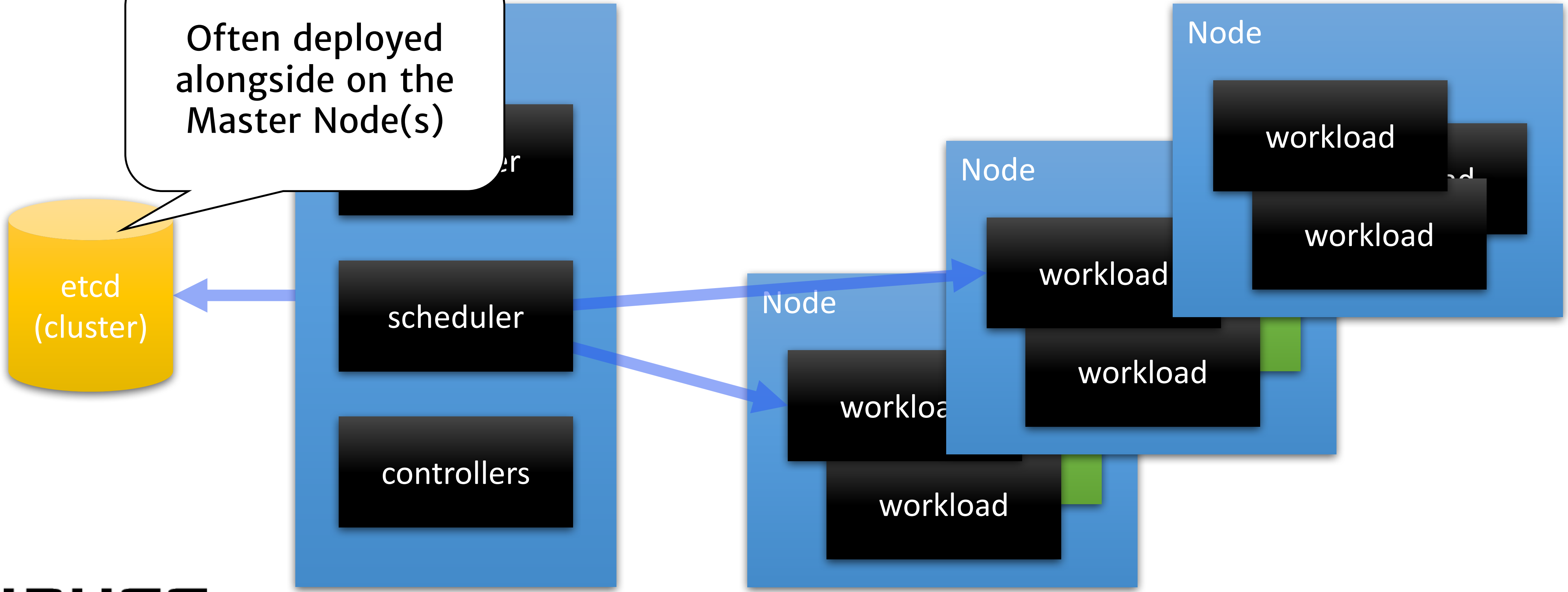
- Provide a runtime environment for Docker containers
- Scale and load balance docker containers
- Abstract away the infrastructure containers run on
- Monitor/health check containers
- Declarative definition for running containers
- Update containers (also rolling updates)
- Storage mounting (allow abstracting infrastructure)
- Service discovery and exposure
- Labelling and selection of any kind of object (we‘ll get to this)

We get to run containers!

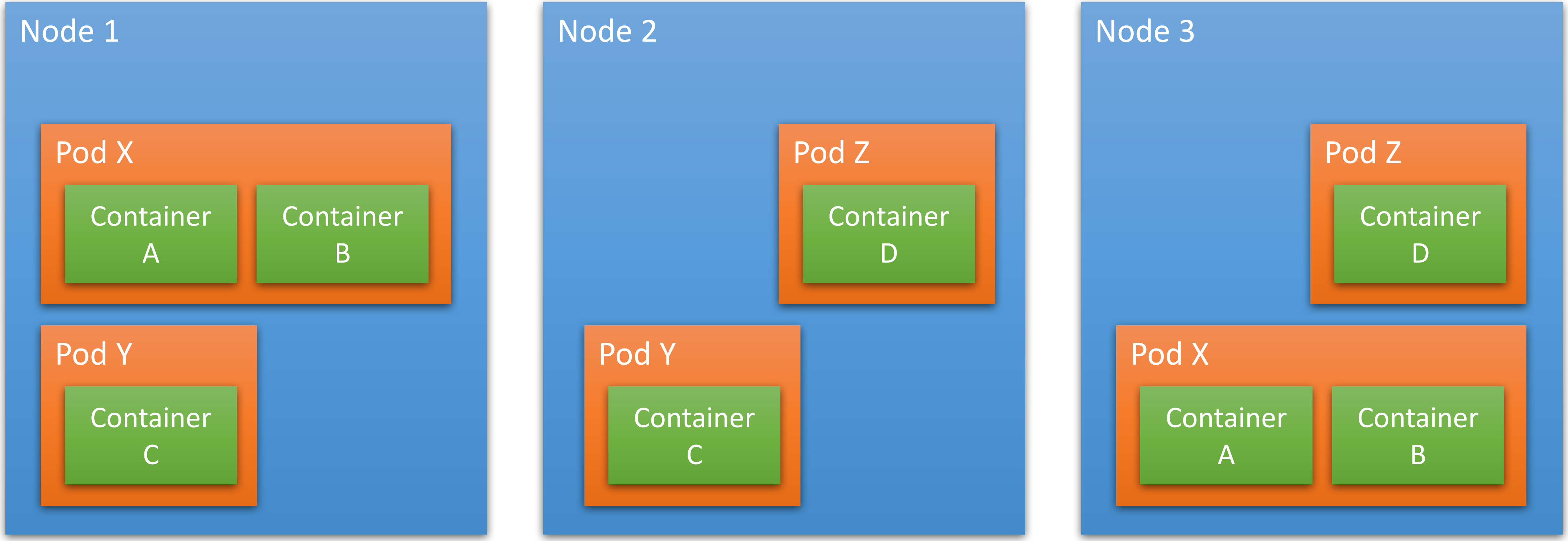
- Kubernetes adds functionality to Docker/Container runtimes (containerd, rkt,...)
- Manages a set of (Docker) Hosts, forming a Cluster
- Takes care of Container scheduling
- Supervises containers
- Kubernetes is an **alternative** to Docker Swarm

Deployment Architecture

All blue boxes are Docker Hosts (VMs)
Kubernetes Components are also running as stateless containers!

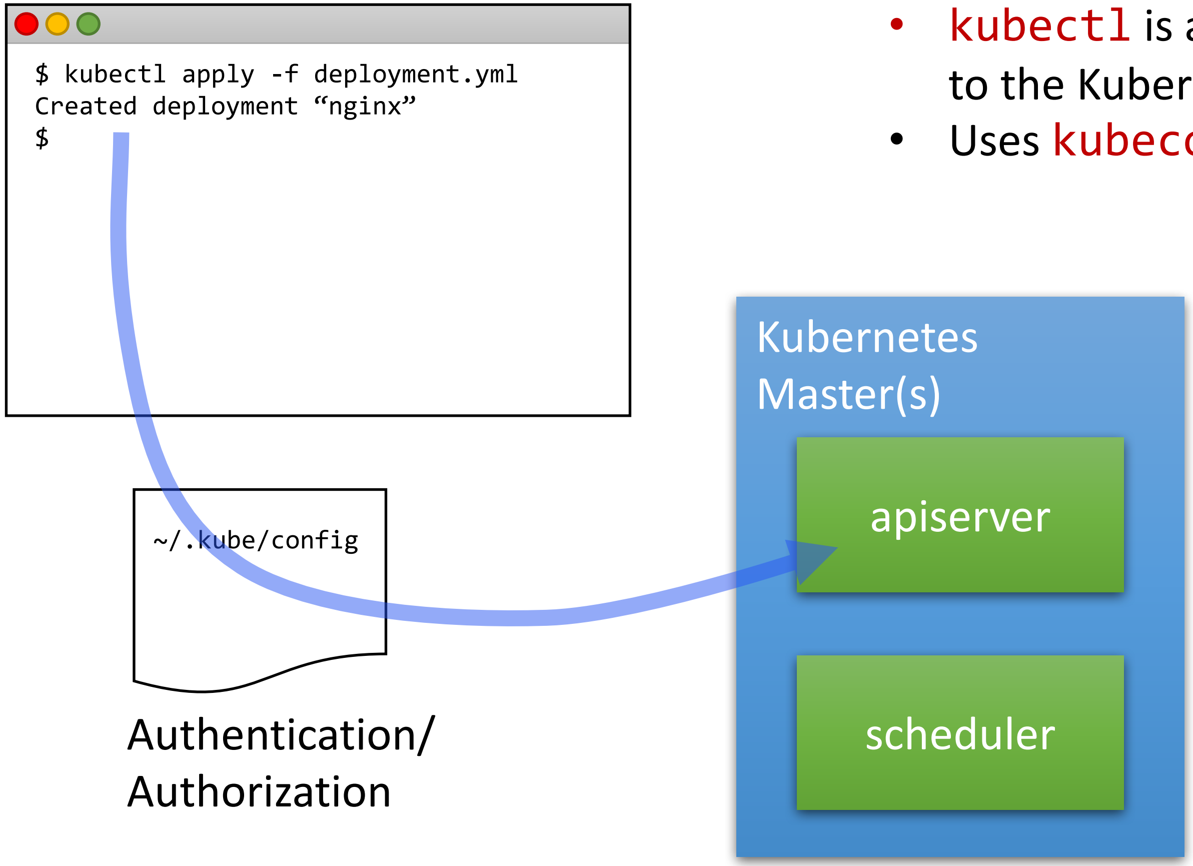


Kubernetes Runtime



Working with kubectl

- kubectl is a convenient way to talk to the Kubernetes API
- Uses kubeconfig for AuthN/Z



```
$ kubectl apply -f deployment.yml
Created deployment “nginx”
$
```

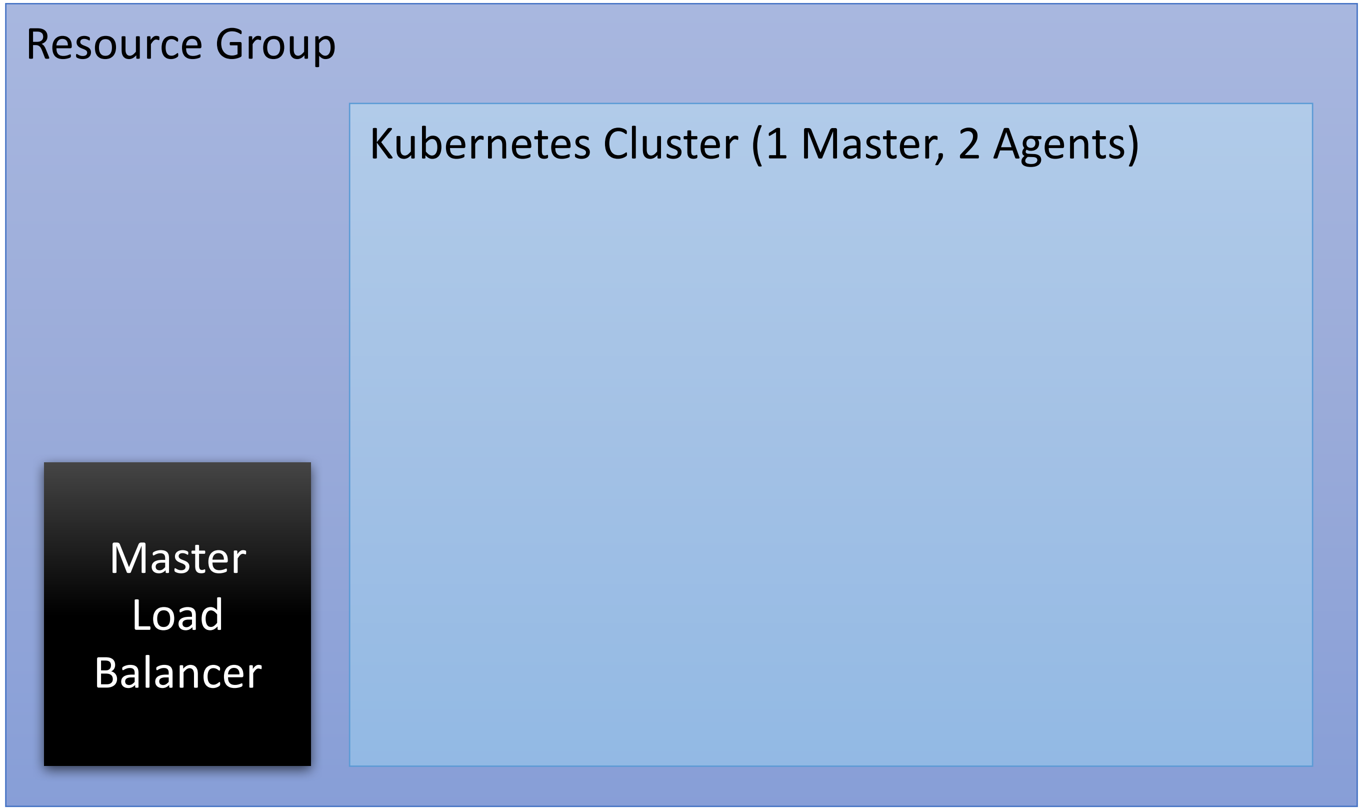
LAB 1
Provision a Cluster

Use Wifi
“HG Mobile”

Lab 1 – Objectives

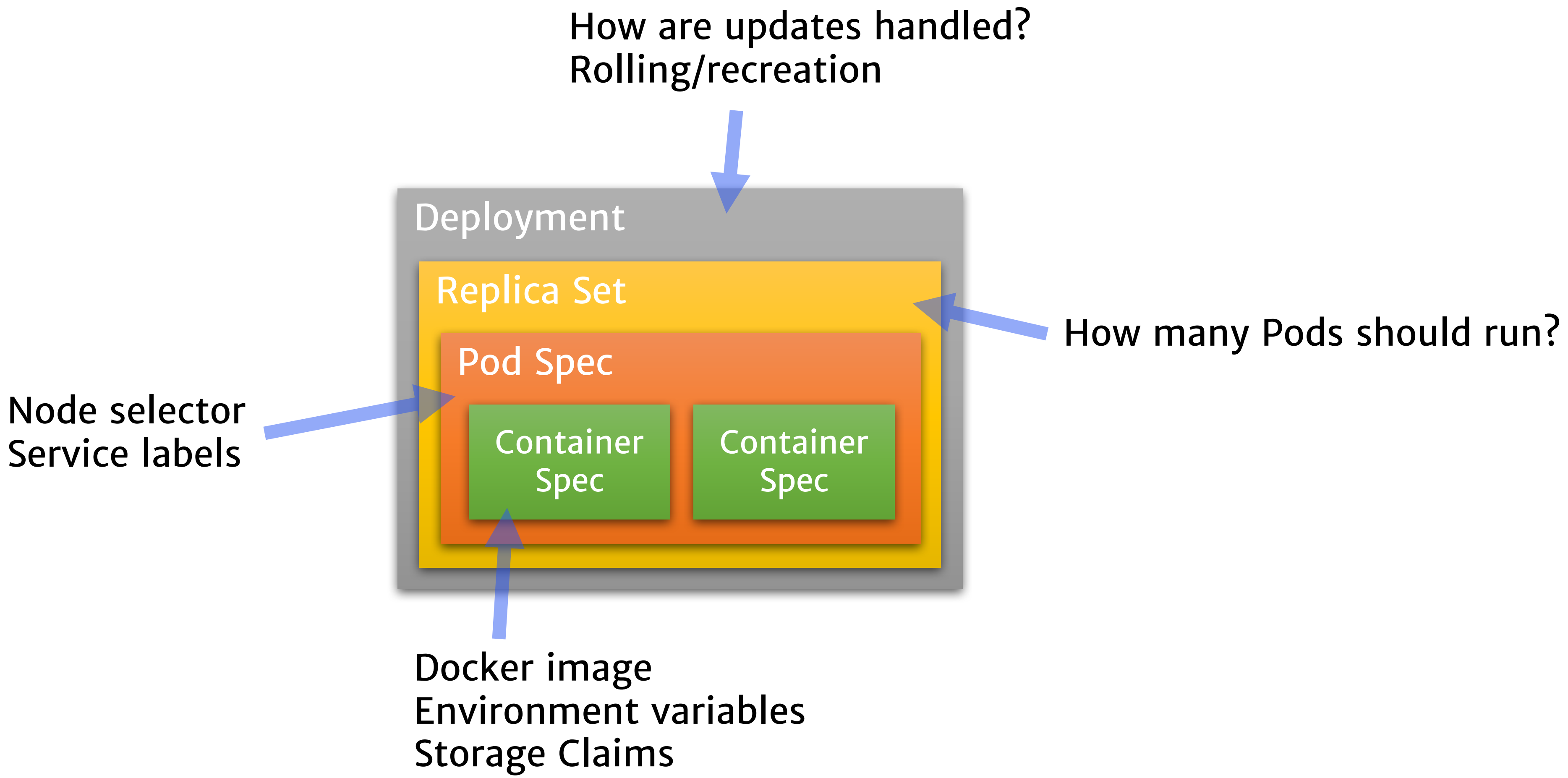
- Make sure you can connect to Azure
- Provision a 1 Master, 2 Agent Kubernetes Cluster
- Install kubectl (Kubernetes CLI)
- Ensure connectivity

State after Lab 1

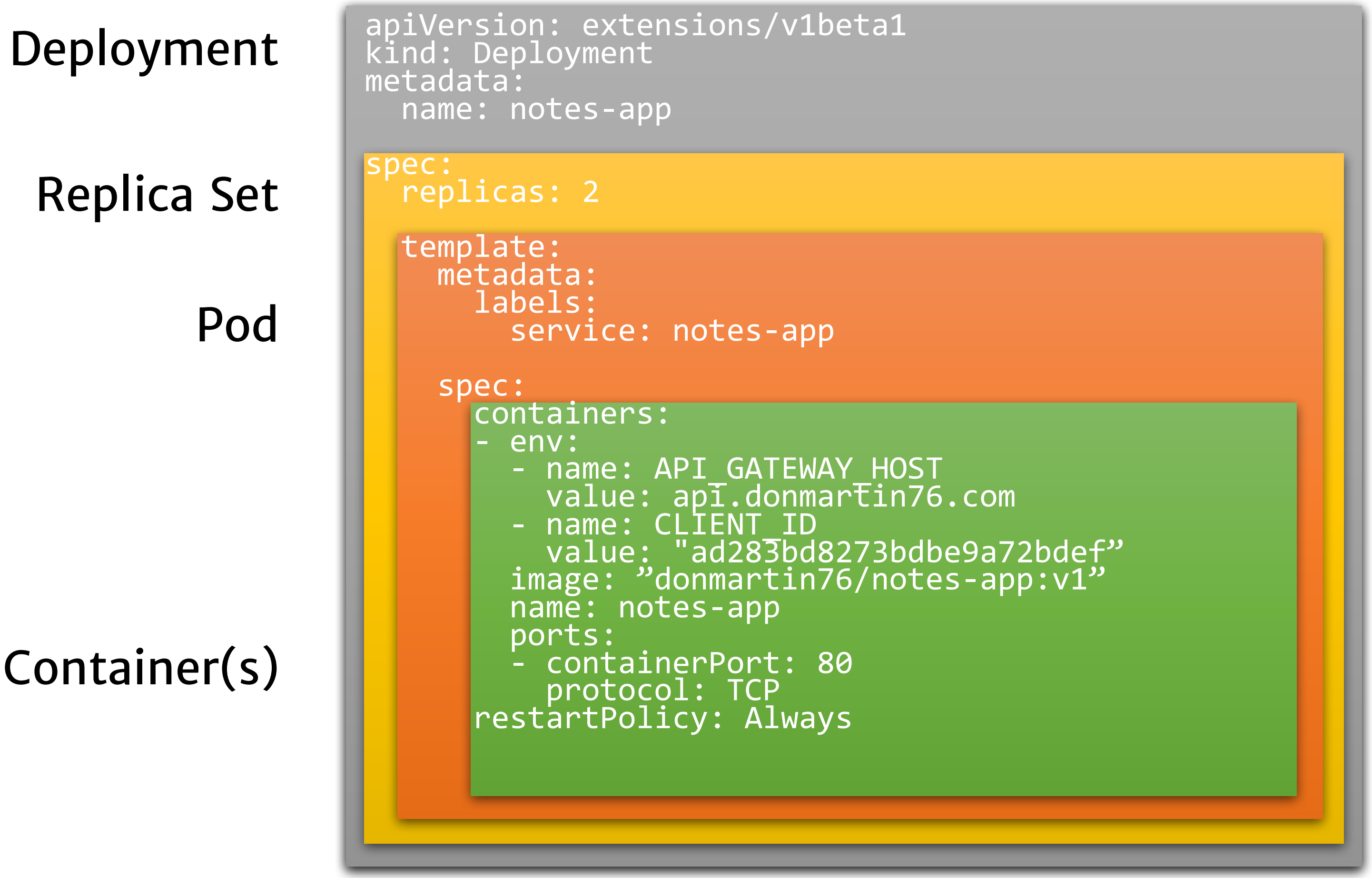


Pods and Deployments

Abstractions – Boxes in Boxes



Example Deployment YAML file



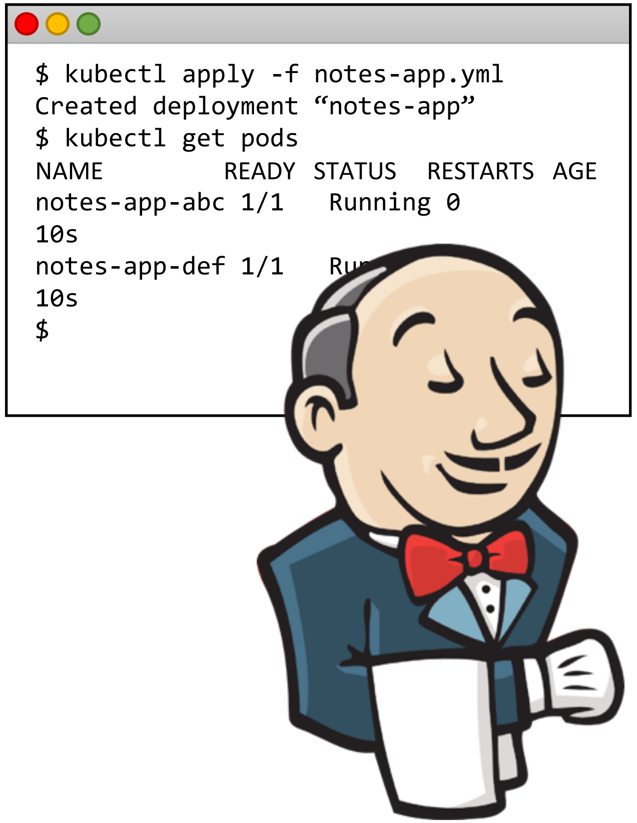
Deployment

Replica Set

Pod

Container(s)

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: notes-app

spec:
  replicas: 2

  template:
    metadata:
      labels:
        service: notes-app

    spec:
      containers:
      - env:
        - name: API_GATEWAY_HOST
          value: api.donmartin76.com
        - name: CLIENT_ID
          value: "ad283bd8273bdbe9a72bdef”
        image: ”donmartin76/notes-app:v1”
        name: notes-app
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
```



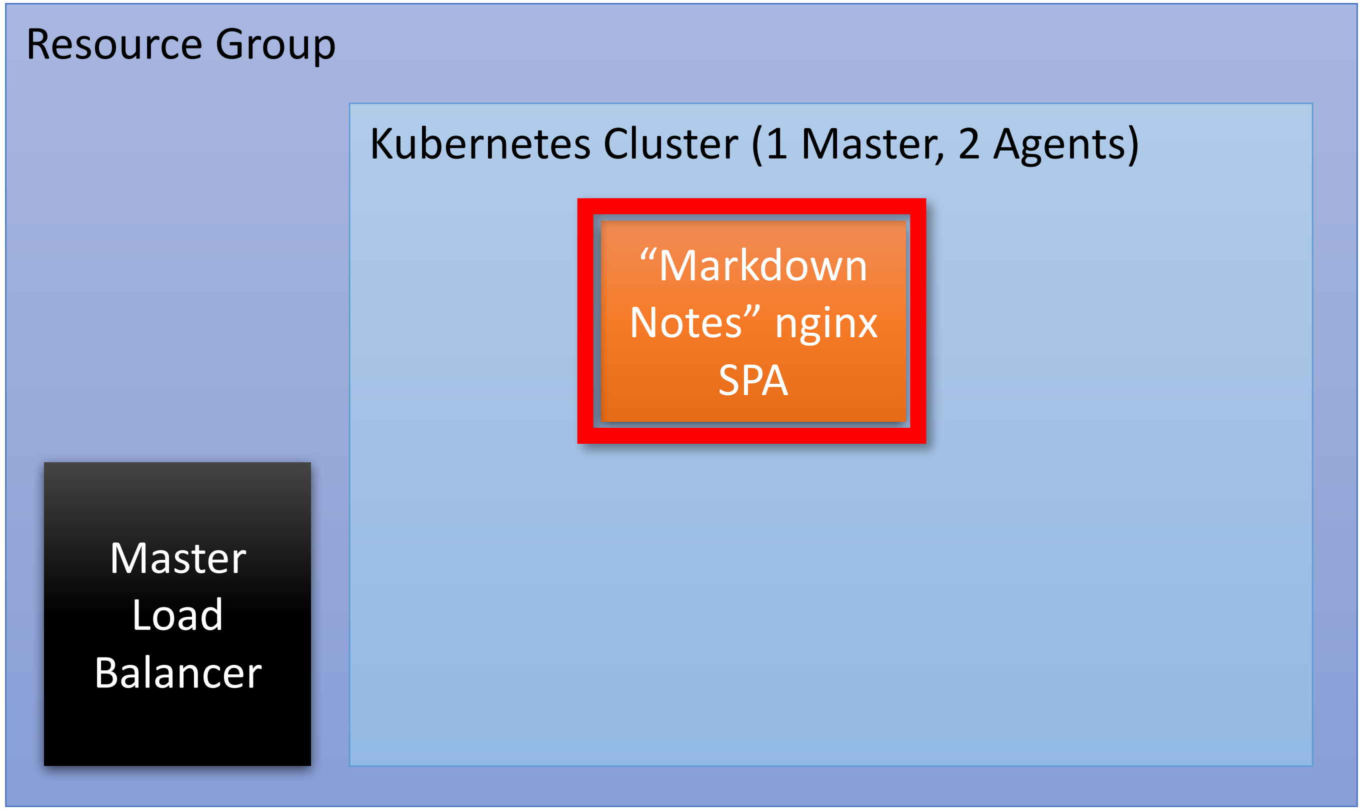
```
$ kubectl apply -f notes-app.yml
Created deployment “notes-app”
$ kubectl get pods
NAME          READY  STATUS   RESTARTS  AGE
notes-app-abc 1/1    Running 0
10s
notes-app-def 1/1    Ru
10s
$
```

Lab 2
Deploy a simple App

Lab 2 – Objectives

- Deploy a simple “Deployment”
- Get some experience with kubectl
- Play whack-a-pod
- Trying out the Kubernetes Dashboard

State after Lab 2

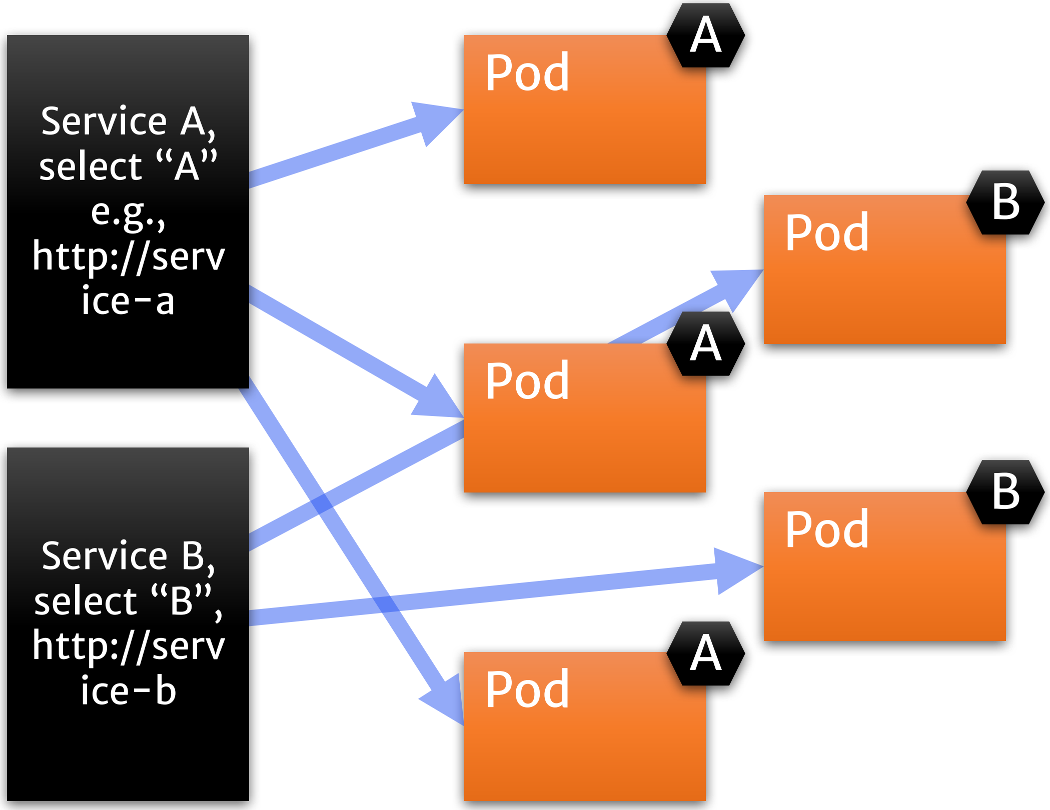

Resource Group
Kubernetes Cluster (1 Master, 2 Agents)
“Markdown Notes” nginx SPA
Master Load Balancer

Services and Ingress

Abstractions – Services

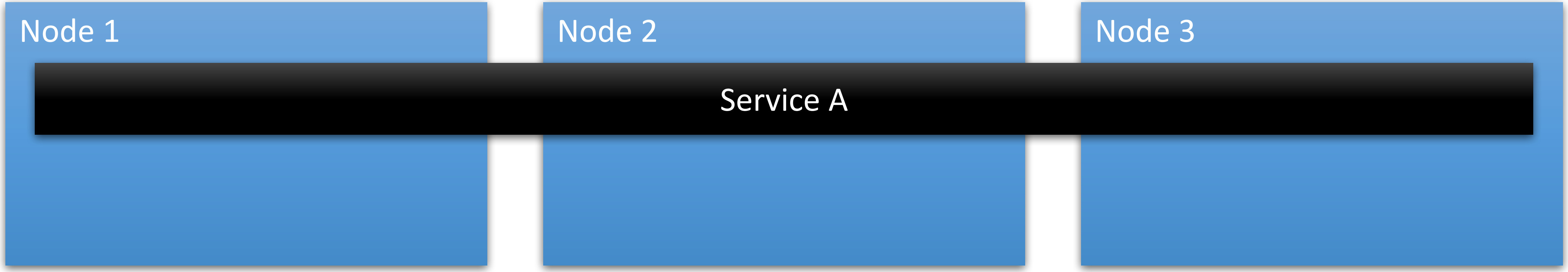
Services …

- Offer discoverability via internal DNS (kube-dns)
- Do automatic pod load balancing
- Can be re-routed dynamically
- Can be defined without backing pods
- Select pods by label matching



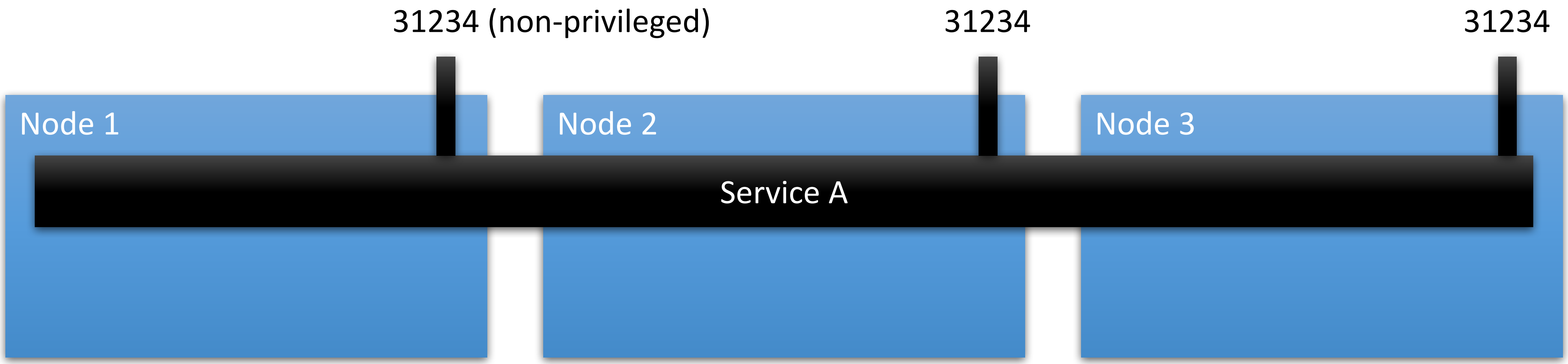
Service A, select “A” e.g., http://service-a
Service B, select “B”, http://service-b
Pod
A
Pod
B
Pod
A
Pod
B
Pod
A

Service Type: Cluster



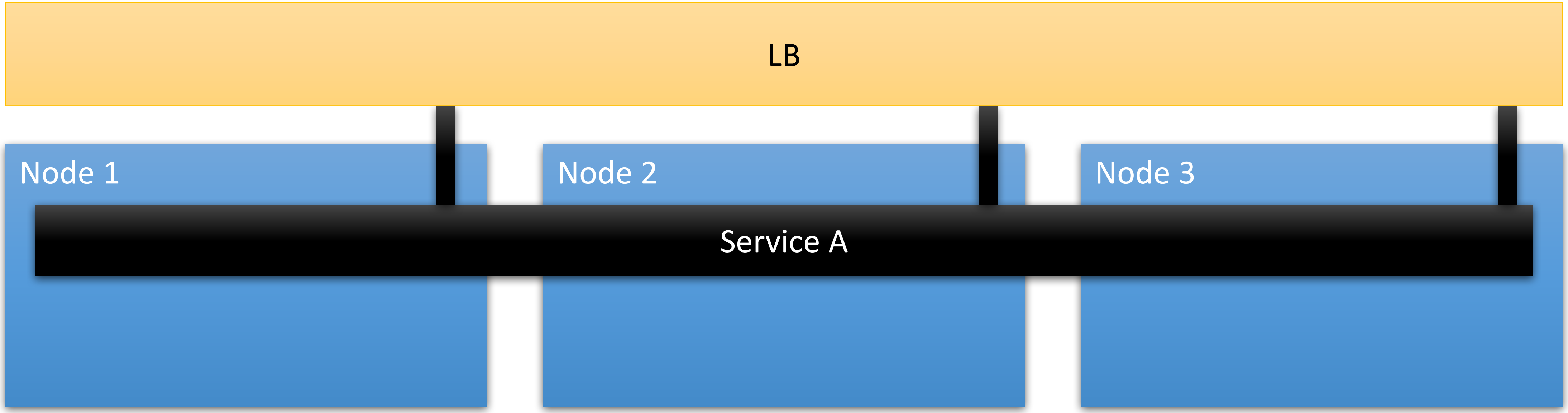
Service can be accessed only from inside the cluster (default mode)

Service Type: NodePort



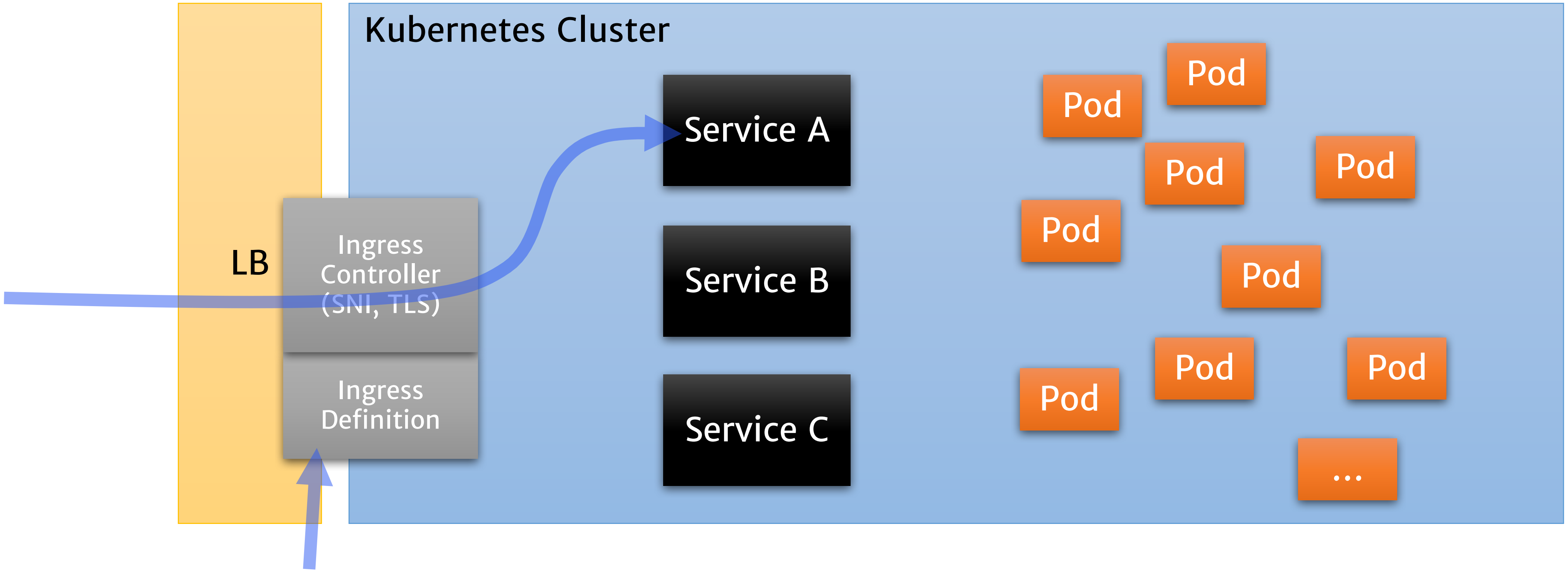
Can be used to manually put an external Load Balancer in front of a service
Common for on-prem clusters leveraging existing load balancers

Service Type: LoadBalancer



Depends on Cloud Provider (Azure, AWS, Rancher,...)
Will provision a Load Balancer with the cloud provider's infrastructure (e.g. Elastic LB, Azure LB,...)
Only works if you really have a cloud provider... ☺

Exposing Services – Ingress



- E.g., “route Host x.y.z to Service A”, “Use TLS Certificate abc for host x.y.z”
- Abstract definition of rules
- Implemented by Ingress Controller
- Flexible; leverages “LoadBalancer” on cloud provider
- Can provide SNI (Server Name Indication) and TLS termination

Example Service YML

```
apiVersion: v1
kind: Service
metadata:
  labels:
    service: notes-app
  name: notes-app
spec:
  type: ClusterIP
  ports:
  - name: “http”
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    service: notes-app
```

Reachable within cluster as
http://notes-app:80

Select pods with this label

Example Ingress YML

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: notes-app
spec:
  rules:
  - host: notes.donmartin76.com
    http:
      paths:
      - path:
        backend:
          serviceName: notes-app
          servicePort: 80
```

Routes to the service with this name and port

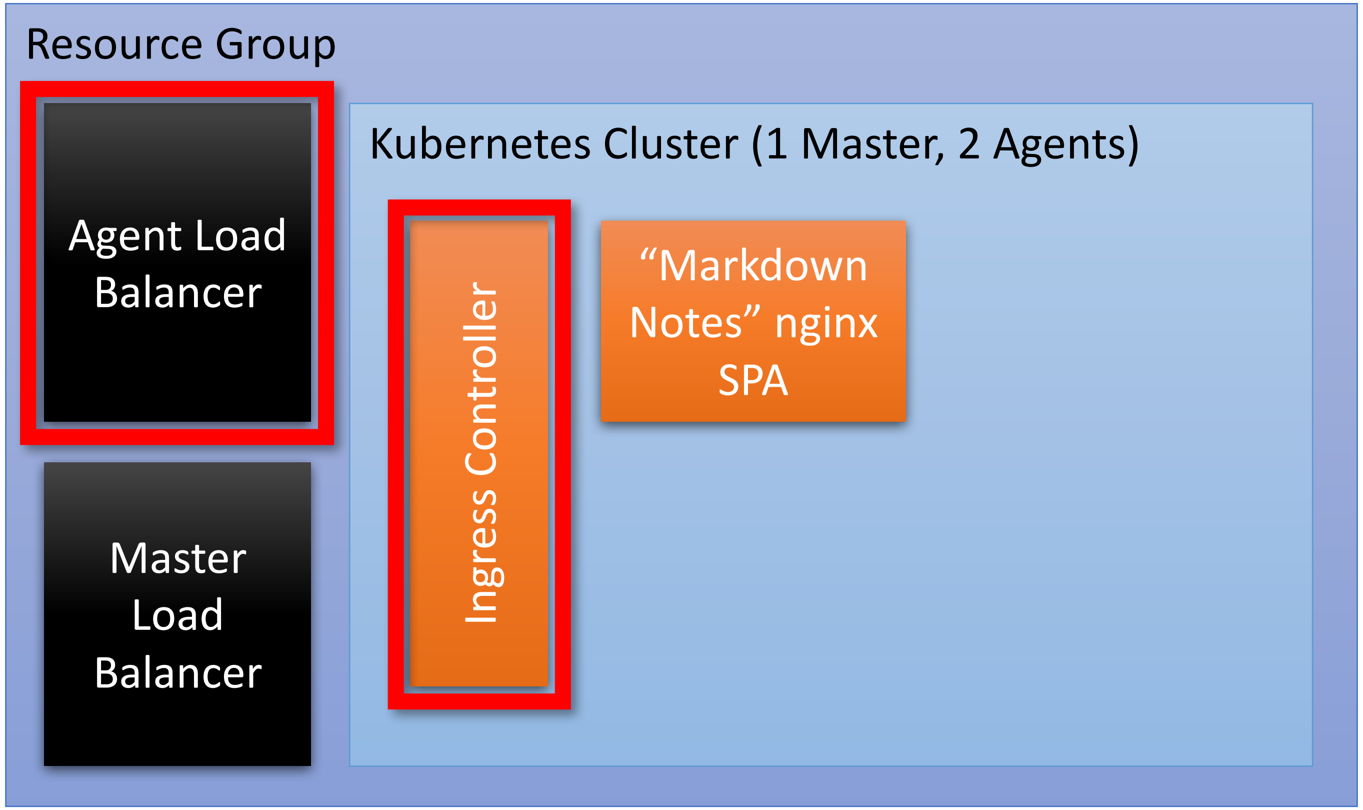
Lab 3
Services and Ingress

HAUFE.

Lab 3 – Objectives

- Deploy a default backend for the cluster
- Create self signed certificates for TLS
- Deploy an nginx Ingress Controller
- Get the load balancer’s IP
- Create DNS entries for our application
- Configure a first ingress resource

State after Lab 3



Resource Group

Agent Load Balancer

Master Load Balancer

Kubernetes Cluster (1 Master, 2 Agents)

Ingress Controller

“Markdown Notes” nginx SPA

Configmaps and Secrets

Configmaps and Secrets

- Stores cluster wide configuration and secrets
- Can be used to inject information to pods
- Useful for externalized configuration
- … and secrets, like credentials
- Usually referred to from within Deployments

```
env:
- name: CLIENT_ID
  valueFrom:
    secretKeyRef:
      name: notes-app-secrets
      key: client_id
```

```
env:
- name: API_HOST
  valueFrom:
    configMapKeyRef:
      name: apim-config
      key: PORTAL_NETWORK_APIHOST
```

Gotchas: Secrets (<=1.7.x)

- Currently, secrets aren’t really secret
- Different resource
  - Almost same mechanism
- Work in progress
- Use with care, only in non-shared-cluster situations

Example ConfigMap YAML

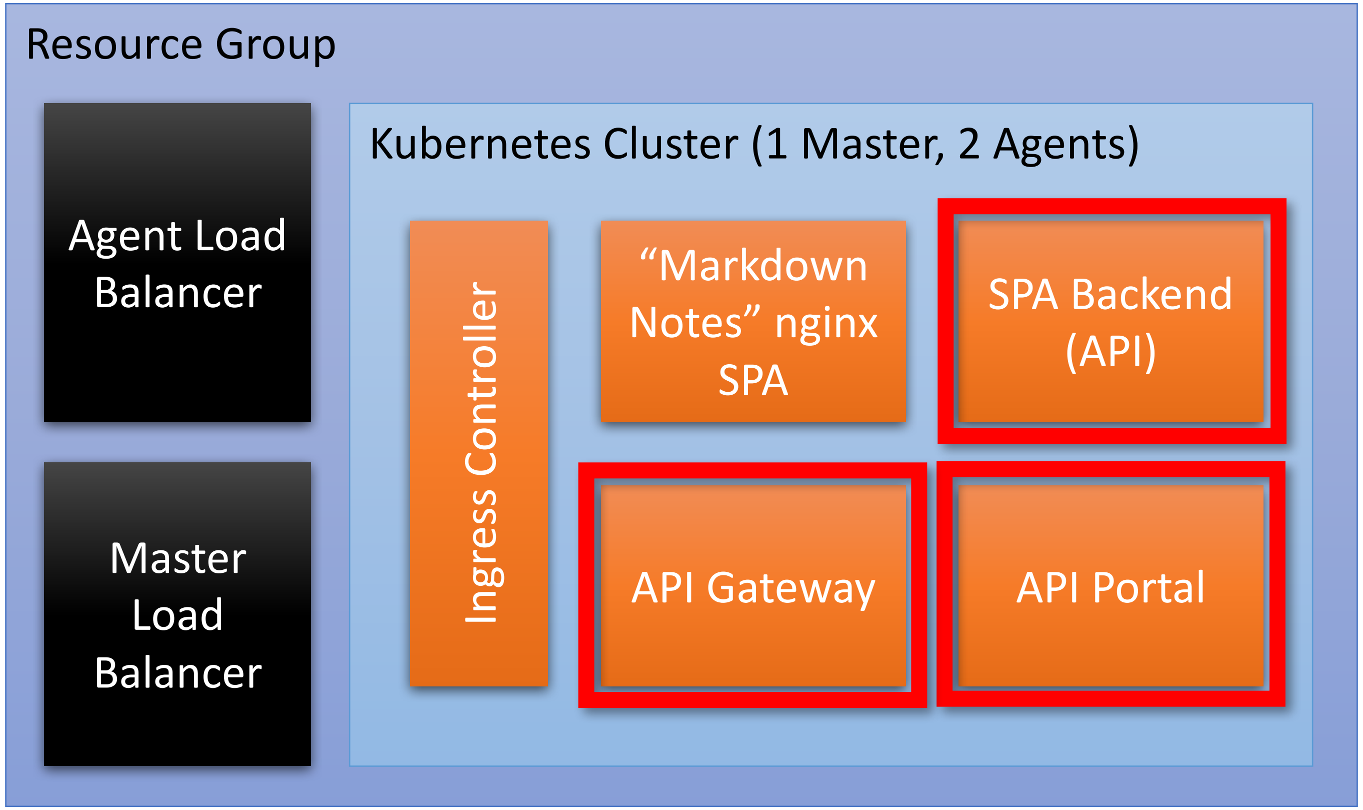
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: apim-config
  namespace: default

data:
  APP_HOST: notes.martin.k8s.donmartin76.com
  GIT_REPO: github.com/DonMartin76/k8s-workshop-apim-config
  PORTAL_NETWORK_APIHOST: api.martin.k8s.donmartin76.com
  PORTAL_NETWORK_PORTALHOST: portal.martin.k8s.donmartin76.com
```

Lab 4
Deploy the full Stack

Lab 4 – Objectives

- Get GitHub client id and secrets for OAuth2 log in
- Add config maps and secrets
- Deploy the rest of the app in one go
- Try out the app

State after Lab 4 (full deployment)



Resource Group

Agent Load Balancer

Master Load Balancer

Kubernetes Cluster (1 Master, 2 Agents)

Ingress Controller

“Markdown Notes” nginx SPA

SPA Backend (API)

API Gateway

API Portal

Lab 5
Scaling and Updating

Lab 5 – Objectives

- Scaling a deployment via command line
- Updating an image via command line
- Demonstration of rolling updates

Topics not covered

App and Cluster Monitoring

- Standard solution: Prometheus for both App and Infra monitoring
- Paired with AlertManager and Grafana
- Additionally do e2e testing from outside cluster (to detect complete failures)
- Could be subject to own workshop/lab

Namespaces & RBAC (1.6+)

- Split workloads into namespaces
- Assign roles to namespaces
- Let specific roles just read, others admin a namespace

Persistent Storage

- Entire deployment is deployed as if stateless
- Corollary: Kill the Notes API and all data is gone
- Kubernetes plays well with
  - NFS
  - GlusterFS
  - CephFS
  - Node storage (dangerous, not recommended)
  - Quobyte
  - ... more, and more are being added
- Nonetheless: No silver bullet for storage (yet) available
  - Aurora uses a self-managed NFS server on Azure – not optimal!

Helm – Kubernetes templates

- “Kubernetes Package Manager”
- What we did with bash – Helm (mostly) does better
  - Deployment templating
  - Standard deployments with slight adaptions
  - Parametrization
- Template sharing, also online (“docker hub” like)
- Upgrading procedures implementable
- Yes, this would be awesome for wicked.haufe.io
- Another level of abstraction (I wanted to spare you for now)

Kubernetes “Operators”

- Components which act as “operators” for service
- E.g., “etcd” operator handles operation of an etcd cluster on Kubernetes
- In the works: “Prometheus” operator, “Postgres” operator
- Typical tasks:
  - Log rotating/pruning
  - Sharding, balancing
  - Scaling out, joining pods to a cluster and vice versa
  - Create new instances of a service (PaaS-like)

Kubernetes API

- Everything (and more) which can be done with kubectl can be done with the API
- Each Pod can (optionally) have access to the API
- This is also how operators (in parts) work
- Self-configuring services, services administrating other services

Lab 6
Cleaning up

Lab 6 – Objectives

- Clean up the mess we made on our subscription

State after Lab 6 (☺)

Wrapup

What’s next?

- Read even more on http://kubernetesbyexample.com
- Roam the documentation at https://kubernetes.io
- Check out Prometheus, it’s complex but cool
- Continue containerizing – Kubernetes is a robust way of running them

That’s all, Folks!

Please fill in the netigate survey, or just give me feedback straight away. Thanks!