

# Kubernetes Introduction Workshop

## Golang Warsaw Meetup

CC BY 2.0

Wojciech Barczynski (wbarczynski.pro@gmail.com)

# Contents

<b>1</b>	<b>Prerequisites</b>	<b>2</b>
1.1	HowTos . . . . .	2
1.2	Verify the setup . . . . .	2
1.3	Kubernetes CLI Basics . . . . .	3
<b>2</b>	<b>Kubectl configuration file</b>	<b>4</b>
<b>3</b>	<b>Exploring Namespace kube-system</b>	<b>5</b>
<b>4</b>	<b>Kubernetes deployments.yaml</b>	<b>7</b>
<b>5</b>	<b>Opening console in your container</b>	<b>9</b>
<b>6</b>	<b>Port-forwarding</b>	<b>10</b>
<b>7</b>	<b>Kubernetes Service</b>	<b>11</b>
<b>8</b>	<b>Modifying kubernetes deployment and service</b>	<b>13</b>
<b>9</b>	<b>Updating service</b>	<b>14</b>
<b>10</b>	<b>Kubernetes Ingress</b>	<b>15</b>
<b>11</b>	<b>Containers vs Pods</b>	<b>16</b>
<b>12</b>	<b>Failover</b>	<b>16</b>
<b>13</b>	<b>How to debug</b>	<b>17</b>
<b>14</b>	<b>Kubernetes configmap</b>	<b>17</b>
<b>15</b>	<b>Kubernetes secret</b>	<b>19</b>
<b>16</b>	<b>Opinionated Configuration</b>	<b>19</b>
<b>17</b>	<b>Kubernetes Persistent Volumes</b>	<b>19</b>
<b>18</b>	<b>Outlook</b>	<b>19</b>

# 1 Prerequisites

You need to feel good with Command Line Interface. You should understand what Docker is.

- Workstation with Linux or OSX recommended.
- Software
  - Minikube
  - Kubernetes CLI
  - Docker
- Tools
  - jq
- Good to have
  - hub.docker.com account or alternative docker repository

## 1.1 HowTos

- Install Minikube at [kubernetes.io](https://kubernetes.io)
- Install Kubernetes CLI - kubectl at [kubernetes.io](https://kubernetes.io)

## 1.2 Verify the setup

```
$ minikube status
```

```
$ minikube start
```

```
$ kubectl config use-context minikube
```

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	master	6d	v1.12.4

## 1.3 Kubernetes CLI Basics

Let's learn first some basics regarding the *kubectl*:

```
kubectl get <ARTIFACT>
kubectl describe <ARTIFACT>
```

1. List the nodes

```
kubectl get nodes
```

What the names of our nodes are? . . .

2. Learn about the node

```
kubectl get nodes minikube
```

# notice:

```
kubectl get no minikube
kubectl get node minikube
```

3. Get more details:

```
kubectl describe nodes minikube
```

Note down:

- Container Runtime Version: . . .
- What the namespaces we have: . . .
- Note down name of two pods:
  - . . .
  - . . .

4. YAML and JSON output

```
kubectl get node minikube -o yaml
kubectl get node minikube -o json
```

Use *jq* to get the *kubeletVersion*, write down below:

...

5. Notice jsonpath support

```
kubectl get node minikube \
  -o jsonpath="{.status.daemonEndpoints.kubeletEndpoint.Port}"

kubectl get node minikube -o jsonpath="{.metadata.labels}"
```

Write down a command with jsonpath to get information on how many CPU we have allocated to our minikube:

...

6. Select with labels

```
kubectl get nodes -l 'kubernetes.io/hostname'=minikube
```

Please find another label, you could select our node and run the command.

7. Select with labels

```
kubectl api-resources
```

Hints

- kubectlx and kubens - <https://github.com/ahmetb/kubectlx>
- alias k=kubectl or alias kb=kubectl

## 2 Kubectl configuration file

Good to know. You can find there also your token, certificates, etc.

1. View `/.kube/config`
2. Find *certificate-authority*
3. Note the main sections:  
.....

### 3 Exploring Namespace kube-system

Let's look around what we have here.

1. What are the namespaces?

```
$ kubectl get ns
$ kubectl get namespaces
```

Notice:

- you can create namespaces to better organize your components
- you define resource restrictions for namespaces
- affect the name: *service-name.namespace-name.svc.cluster.local*.  
We will talk about it later.

To change the selected namespace for our commands

```
kubectl config set-context \
$(kubectl config current-context) \
--namespace <namespace-name>
```

2. Get the list of pods in namespace kube-system:

```
$ kubectl get po --namespace=kube-system
$ kubectl get po -n=kube-system
```

Use `kubectl describe po <pod-name> --namespace=kube-system` to find what the version is of:

- kube-proxy: . . .
- apiserver: . . .
- coredns: . . .

3. Get the list of services:

```
$ kubectl get svc --namespace=kube-system
```

Use `kubernetes describe svc <svc-name> --namespace=kube-system` to find the endpoints for:

- kube-dns: . . .
- kubernetes-dashboard: . . .

#### 4. Logs:

```
$ kubectl logs coredns-c4c -n=kube-system
$ kubectl logs coredns-c4c -n=kube-system -f
$ kubectl logs coredns-c4c -n=kube-system --tail=10
```

Please display logs of:

kube-apiserver, kube-proxy, kube-scheduler, and etcd-minikube.

Later, we will also cover events:

```
kubectl get events -n=kube-system.
```

#### 5. Get the console:

```
$ kubectl exec -it kube-apiserver-minikube \
/bin/sh -n=kube-system
```

#### 6. Kubernetes Dashboard:

```
# on normal deployment:
# $ kubernetes proxy
$ minikube dashboard
```

#### 7. Basic metrics:

```
minikube addons enable metrics-server
```

```
# wait 5 seconds
kubectl top nodes
kubectl top pods
```

## 4 Kubernetes deployments.yaml

We have been playing around with kube-system, now let's get our service up and running.

1. Get the deployment file introduction/kube-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: intro-app
  labels:
    app: intro-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: intro-app
  template:
    metadata:
      labels:
        app: intro-app
    spec:
      containers:
        - name: app
          image: wojciech11/api-status:1.0.0
          ports:
            - containerPort: 80
```

Notice:

- if your repo is private, you need to define `imagePullSecrets`.
- A force pulling image every time: `imagePullPolicy: Always`, helpful during development, do not use in **production**.

2. Deploy:

```
$ kubectl create -f introduction/kube-deployment.yaml
```



3. Check:

```
$ kubectl get po
$ kubectl get po -n default

$ kubectl describe po <POD_NAME>
```

4. To learn about the deployments:

```
$ kubectl get deploy
$ kubectl get deployment

$ kubectl describe deploy <DEPLOYMENT_NAME>
```

What the update strategy do we use?

. . .

5. Find the following information:

- What is the IP of your app pod? . . .
- What is ReplicatSet? . . .
- Ready? . . .
- Restart Count? . . .
- Events? . . .

Notice: the next step are Liveness/Readiness probes.

## 5 Opening console in your container

Imagine, we cannot reach our application. Let's check it within the running container.

1. Get the console:

```
kubectl exec -it intro-app-65d /bin/bash
```

Notice: There is an ongoing discussion whether it is the "new" ssh.

2. Add tool for debugging - curl:

```
apt-get update && apt-get install -qq curl
```

3. Does it work?

```
# does it work?
```

```
curl 127.0.0.1
```

```
# can we get outside
```

```
curl -I wbarczynski.pl
```

```
# can we reach other services:
```

```
telnet kube-dns.kube-system 53
```

4. Yes, we fix the bug, let's clean up:

```
kubectl delete po intro-app-65db4-...
```

```
# notice the name change
```

```
kubectl get po
```

## 6 Port-forwarding

What If I told you, you can debug your app from your laptop.

1. Find the port our service listen, check the deployment file.
2. Setup the port forwarding:

```
kubectl proxy-forward <POD_NAME> 8080:<PORT_NUMBER>
```

3. Use curl to query our app from your local console.

Let's learn about services and ingresses first, later we see how we can modify our deployment and update the application.

## 7 Kubernetes Service

Our factory, I mean the deployment defines how we create our applications as pods. The service, how it is consumed.

1. Get the service file `introduction/kube-service.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: intro-app
  labels:
    app: intro-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: intro-app
  template:
    metadata:
      labels:
        app: intro-app
    spec:
      containers:
      - name: app
        image: wojciech11/api-status:1.0.0
        ports:
        - containerPort: 80
```

2. Deploy:

```
$ kubectl create -f introduction/kube-service.yaml
$ kubectl apply -f introduction/kube-service.yaml
```

3. Check:

```
$ kubectl get svc
```

4. Find out the endpoint IP: . . .

5. We few types of services:

- with ClusterIP
- ClusterIP: None
- LoadBalanced

6. Let's access it:

```
export SVC_PORT=$(kubectl get service intro-app \
  --output='jsonpath="{.spec.ports[0].nodePort}"' \
  | tr -d ' ')
curl $(minikube ip):${SVC_PORT}
```

Notice: on Azure, AWS, or Google, we would get the loadbalancer created and public IP assigned.

7. How it works?

```
$ kubectl exec -it intro-app-65db487447-lrhh9 /bin/bash
```

```
/# curl intro-app
/# curl intro-app.default
/# curl intro-app.default.svc
```

```
/# apt-get update && apt-get install dnsutils -qq
/# # resolve the same host names:
/# nslookup intro-app
```

8. How it works from other apps?

```
$ kubectl run -i --tty busybox --image=busybox -- sh
/# curl intro-app
/#
/# wget -O- intro-app
/# wget -O- kubernetes-dashboard.kube-system
```

## 8 Modifying kubernetes deployment and service

Avoid editing files on kubernetes, always modify a yaml and apply the changes.

1. Change the number of pods running to 2 with:

```
$ kubectl edit deploy
```

```
$ kubectl get po
```

2. Change the value of label `me` to your name in the service definition.
3. Modify the `deployment.yml` to get 3 running pods, use: `kubectl apply -f`
4. Add one more label to service.
5. What does happen if we add one more selector, apply it:

```
apiVersion: v1
kind: Service
metadata:
  name: intro-app
  labels:
    me: wojtek
spec:
  ports:
    - port: 80
      protocol: TCP
  selector:
    app: intro-app
    break: the-connection-with-pods
  type: LoadBalancer
```

Can we connect?

```
curl -I $(minikube ip):${SVC_PORT}
```

What has changed?

```
kubectl describe svc intro-app
```

Notice: very very very common issue :D

6. Fix our service.

## 9 Updating service

Let's update our app from the version 1.0.0 to 2.0.0:

1. Change in the deployment file and apply changes.
2. You can also change it with set image:

```
$ kubectl set image deployment/<CONTAINER_NAME> \  
  <CONTAINER_NAME>=<DOCKER_IMAGE_NAME>:<VERSION>
```

3. Change two times from 1.0.0 to 2.0.0 and back:

```
$ curl -I $(minikube ip):${SVC_PORT}
```

## 10 Kubernetes Ingress

1. Install our ingress-controller:

```
$ kubectl apply -f introduction/ingress/
```

2. Apply the ingress controller:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: intro-app
  annotations:
    kubernetes.io/ingress.class: traefik
    traefik.frontend.rule.type: PathPrefixStrip
spec:
  rules:
  - host: my.app
    http:
      paths:
      - path: /echo
        backend:
          serviceName: intro-app
          servicePort: 80
```

3. Check:

```
kubectl get ing
kubectl describe ing <YOUR_ING>
```

4. Use:

```
$ export ING_CNTR_PORT=$(kubectl \
get service traefik-ingress-service \
-n kube-system \
--output='jsonpath="{.spec.ports[0].nodePort}"' \
| tr -d '')
```



```
$ export MK_IP=$(minikube ip)
```

```
$ curl --header 'Host: my.app' http://${MK_IP}:${ING_CNTR_PORT}/echo
```

5. Open traefik dashboard:

```
kubectrl port-forward -n kube-system \
traefik-ingress-controller-s58cv \
8080:8080
```

## 11 Containers vs Pods

Please answer the following questions:

- How many containers can a Pod has?
- Do containers share disk?
- Do container share port space?
- What does 1/1 mean in the output of `kubectrl get po`?

## 12 Fail-over

1. Open console.

2. Force restart:

```
# should work
kill 1
```

```
# always works
kill -9 1
```

Repeat 5 times. Observer the output from: `kubectrl get po`.

## 13 How to debug

Good to ship a minimum of debugging tools in your container, such as, curl or telnet.

Happy debugging path:

```
kubectl describe ing
kubectl describe svc
kubectl exec -it <> /bin/bash
```

```
# curl, telnet, ...
kubectl describe po <>
```

```
kubectl logs <>
kubectl logs <> -f
kubectl logs <> --tail=100
```

```
kubectl logs <YOUR_INGRESS_CONTROLLER_POD>
```

```
kubectl get events
```

## 14 Kubernetes configmap

With configmaps, we can deliver values for environment variables or files. Let's change the page in our application:

1. Copy index.html:

```
cp introduction/dockers/site-1.0.0/index.html index.html
```

2. Add your name after the version number:

```
<html>
<h1>1.0.0-Maria</h1>
</html>
```

3. Let's create a configmap:

```
kubectl create cm index-html --from-file index.html
```

4. Check the commands:

```
kubectl describe cm index-html
kubectl get cm index-html -o yaml
kubectl get cm index-html -o json
```

5. Let's mount it:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: intro-app
  labels:
    app: intro-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: intro-app
  template:
    metadata:
      labels:
        app: intro-app
    spec:
      containers:
        - name: app
          image: wojciech11/api-status:1.0.0
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: "html-content"
      volumes:
        - name: html-content
          configMap:
            name: index-html
```

5. Smoke test:

```
# 30288 is the port, the ingress-controller is listening:
curl --header 'Host: my.app' "http://$(minikube ip):30288/echo"
```

## 15 Kubernetes secret

The same as the Kubernetes config, difference, it is stored only on the master nodes.

## 16 Opinionated Configuration

1. envsubst
2. Ksonnect
3. Helm

## 17 Kubernetes Persistent Volumes

A persistence storage that survives your pod being deleted.  
TBD: mention about empty

## 18 Outlook

What could be the next steps in learning k8s.

What you could learn next.

Next course *Immediate (Developer)*:

1. Liveness/Readiness probes
2. Monitoring with Prometheus
3. Resource and Limits, QoS for your pods, schedule policies
4. Statefulsets
5. DaemonSets

*Observability plus* with Istio demo as what might the future be:

1. Monitoring
2. Logging
3. Traceability

*Advance (Developer):*

1. Zero-downtime deployment strategies
2. Horizontal scaling (beta: vertical pod scaling for the pets)
3. Continuous Deployment and Integration
4. TravisCI and Gitlab
5. (sic :/ ) gitops, have to say few things...

*Network and Security:*

1. RBAC deep dive
2. Networking - Internal Loadbalancing - <https://kubernetes.io/docs/concepts/services-networking/>
3. Egress - <https://kubernetes.io/docs/concepts/services-networking/network-policies/>
4. IpBlock

*Kubernetes customization*

1. Write your first CRD
2. ...
3. ...
4. FaaS? Kubeless?

*CloudNative Ecosystem*

1. Observability: Prometheus stack
2. Observability: EFK
3. Observability: Tracing
4. Ingress Controllers: Traefik, ... , talk about standard and controller-specific annotations
5. Cert-manager
6. (Traditional) backups with Ark
7. Operators for etcd and Vault

*More*

1. Istio
2. Operators for ...

*Optionals*

1. Google Kubernetes Engine - GKE
2. Azure Kubernetes Service - AKS
3. Amazon Elastic Kubernetes - EKS

Contact: [wbarczynski.pro@gmail.com](mailto:wbarczynski.pro@gmail.com)