
Kubernetes Setups
redhat.

Kubernetes setups

- Kubernetes should really be able to run **anywhere**
- But, there are more **integration** for certain cloud providers, like AWS & GCE
  - Things like **Volumes** and **External Load Balancers** work only with supported Cloud Providers
- I will first use **minikube** to quickly spin up a local single machine with a Kubernetes cluster
- I'll then show you how to spin up a multi-node cluster

- Later : AWS using Kops***

Kubernetes setups

- Doing the lab yourself is possible (and highly recommended):
  - You can use AWS Free tier account for the same.
  - Using your local machine
    - https://github.com/kubernetes/minikube

Minikube setups

- **Minikube** is a tool that makes it easy to run Kubernetes locally
- Minikube runs a single node kubernetes cluster inside Linux VM
- It's aimed on users who want to just test it out or use it for development
- It cannot spin up a production cluster, it’s a one node machine with no high availabily

Minikube setups

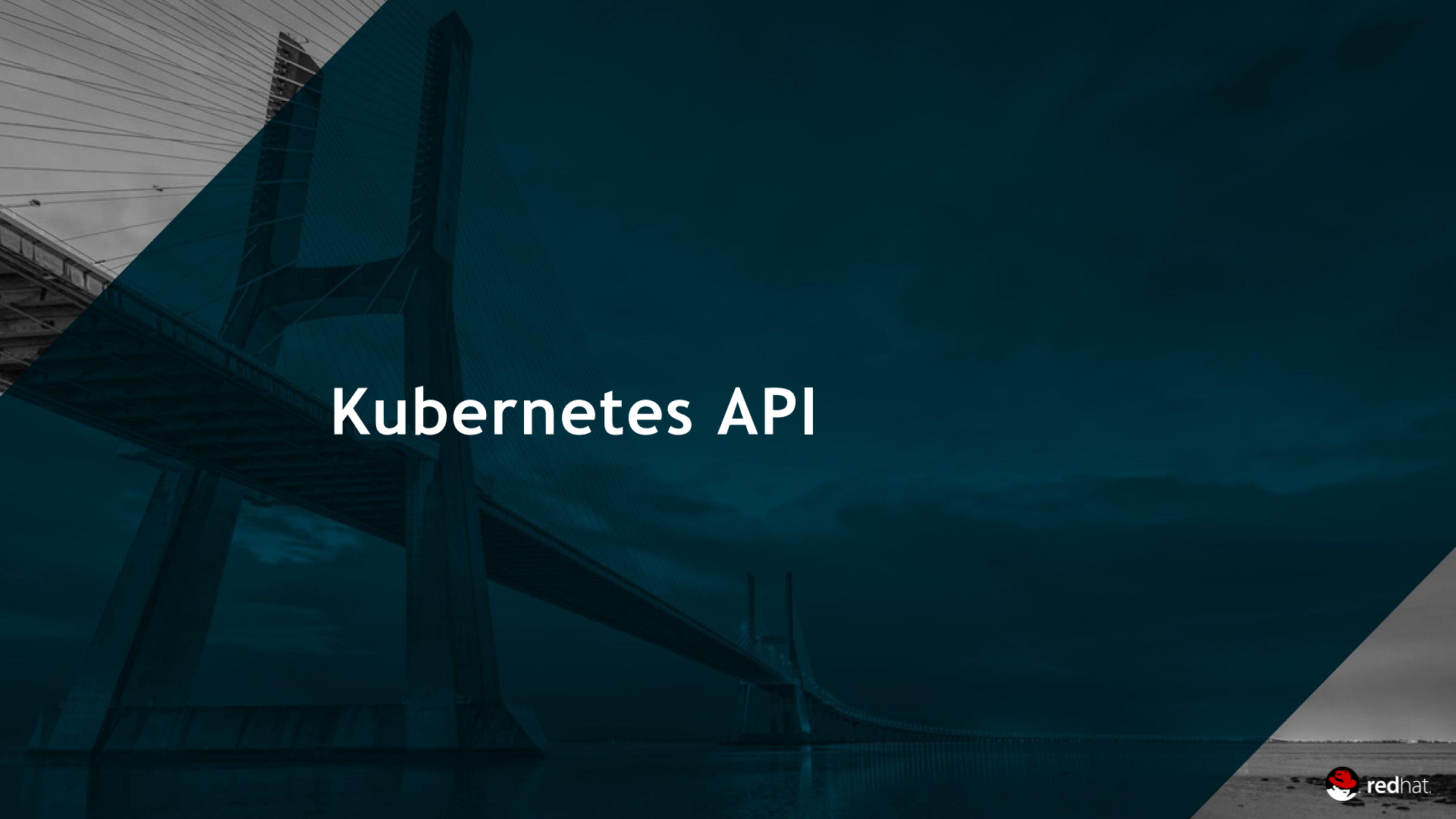
- It works on Windows, Linux and MacOS
- You will need Virtualization Software installed to run Minikube:
  - VirtualBox is freeware
- You can download minikube from

  https://github.com/kubernetes/minikube
- To launch your cluster you just need to enter (in a shell / powershell);

```
.\minikube.exe start --vm-driver="virtualbox"
```

Demo Placeholder

Local kubernetes setup using minikube


Kubernetes API
redhat.

Kubernetes API

- The Kubernetes API also serves as the foundation for the declarative configuration schema for the system.
- The kubectl command-line tool can be used to create, update, delete, and get API objects.

OpenAPI and Swagger definitions

- The Kubernetes apiserver (aka “master”) exposes an API
- Starting with Kubernetes 1.10, OpenAPI spec is served in a single endpoint.
- Swagger has been deprecated and will get removed in Kubernetes 1.14.

Examples of getting OpenAPI spec:

Before 1.10	Starting with Kubernetes 1.10
GET /swagger.json	GET /openapi/v2 **Accept**: application/json
GET /swagger-2.0.0.pb-v1	GET /openapi/v2 **Accept**: application/com.github.proto-openapi.spec.v2@v1.0+protobuf
GET /swagger-2.0.0.pb-v1.gz	GET /openapi/v2 **Accept**: application/com.github.proto-openapi.spec.v2@v1.0+protobuf **Accept-Encoding**: gzip

API Versioning

- Kubernetes supports multiple API versions, each at a different API path, such as **/api/v1** or **/apis/extensions/v1beta1**.

- We chose to version at the API level rather than at the resource or field level to ensure that the API presents a clear, consistent view of system resources and behavior, and to enable controlling access to end-of-lifed and/or experimental APIs.

API Versioning

Different API versions imply different levels of stability and support.

Alpha level

- The version names contain alpha (e.g. v1alpha1).
- May be buggy. Enabling the feature may expose bugs. Disabled by default.
- Support for feature may be dropped at any time without notice.

Beta level

The version names contain beta (e.g. v2beta3).

Code is well tested. Enabling the feature is considered safe. Enabled by default.

Support for the overall feature will not be dropped, though details may change.

API Versioning

Stable level

- The version name is vX where X is an integer.
- Stable versions of features will appear in released software for many subsequent versions.

API Groups

To make it easier to extend the Kubernetes API, API groups implemented

- The API group is specified in a REST path and in the **apiVersion** field of a serialized object.

Currently there are several API groups in use:

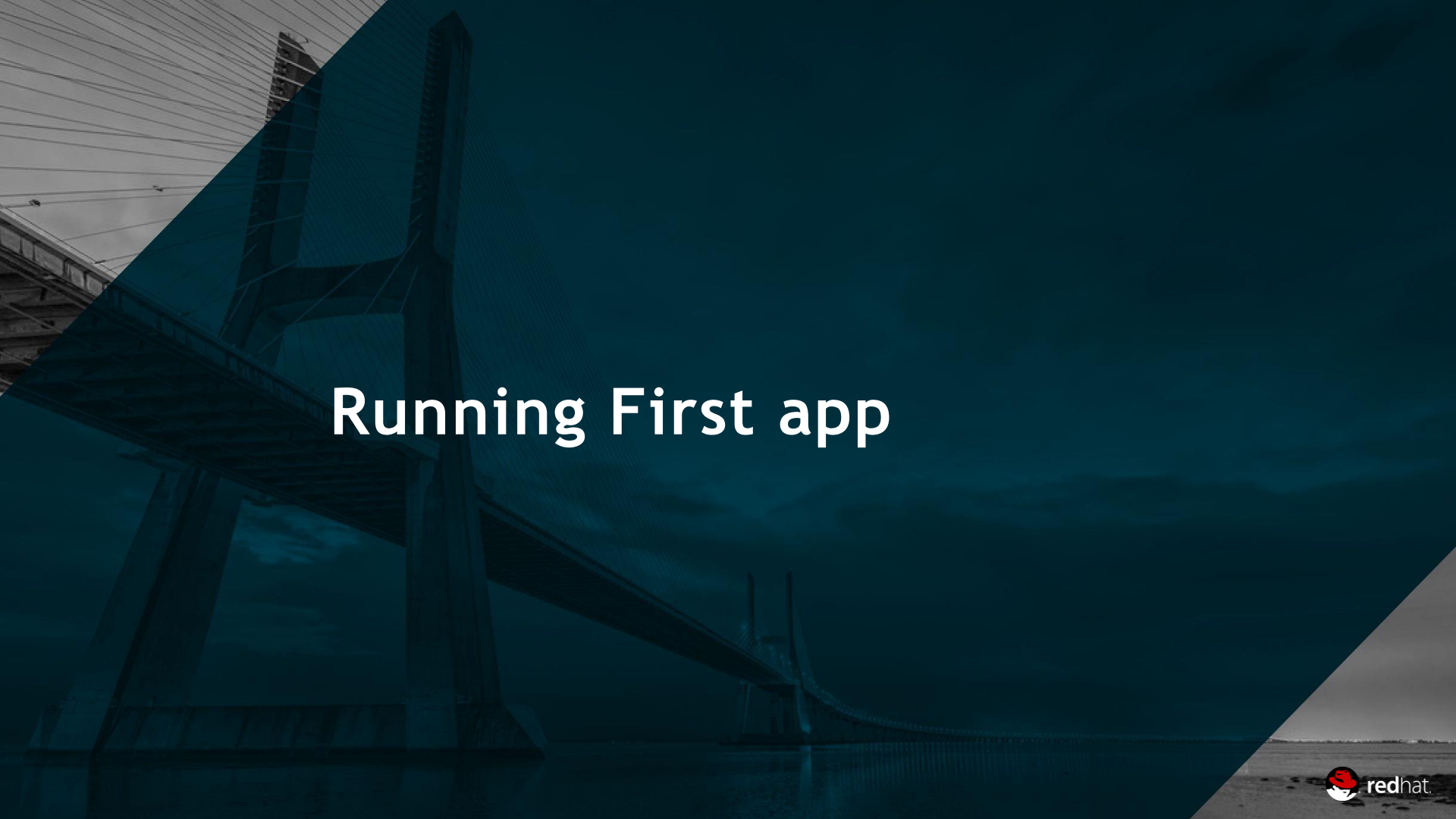
- The core group, often referred to as the legacy group, is at the REST path **/api/v1** and uses **apiVersion: v1**.
- The named groups are at REST path **/apis/$GROUP_NAME/$VERSION**, and use **apiVersion: $GROUP_NAME/$VERSION** (e.g. **apiVersion: batch/v1**). Full list of supported API groups can be seen in Kubernetes API reference.

Enabling API groups

We can be enabled or disabled by setting **--runtime-config** on apiserver**. --runtime-config** accepts comma separated values.

Example

- To disable batch/v1, set --runtime-config=batch/v1=false
- To enable batch/v2alpha1, set --runtime-config=batch/v2alpha1


Running First app
redhat.

First app

- Let’s run our newly built application on the new Kubernetes cluster
- Before we can launch a container based on the image, we need to create a **pod definition**
- **A pod** describe an application running on kubernetes
- A pod can contain one or more tightly coupled containers, that make up app
  - Those apps can easily communicate with each other using their local **port numbers**
- Our app only has **one** container

Create a pod

- Create a file pod-helloworld.yml with pod definition:

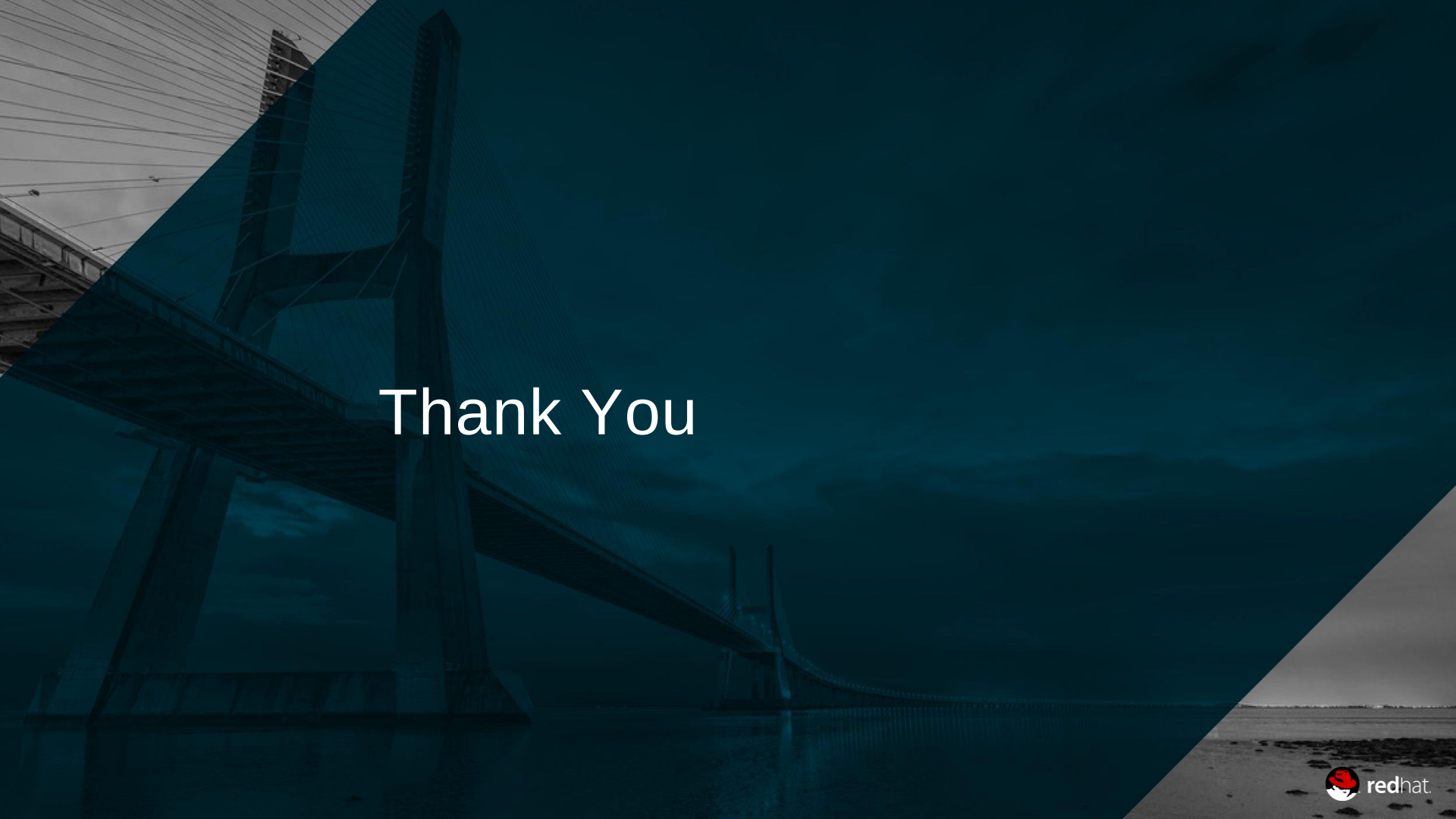
```
apiVersion: v1
kind: Pod
metadata:
  name: nodehelloworld.example.com
  labels:
    app: helloworld
spec:
  containers:
  - name: k8s-demo
    image: amitvashist7/k8s-tiny-web
    ports:
    - name: nodejs-port
      containerPort: 80
```

- Use kubectl to create the pod on kubernetes cluster:

```
.\kubectl.exe create -f first-app/helloworld.yml
```

Demo Placeholder

Running first app on kubernetes


Thank You
redhat.