Simple API With NodeJS

NodeJS Simple API built on Node+Express With MongoDB

This API responds to POST requests at http:///app with a user _id & timestamp. These timestamps are stored in a persistent MongoDB Database. IP addresses & User Id's can be retrieved via HTML, by making a GET request.

(You can use your web browser for this to see all POST's) simply point your browser to http:///app as a get request. The data will be displayed.

Scalability.

This project is deployed on Kubernetes using the included YAML & Secret files.

I have included the .env file for port/DB name details. There is load balancing on the Database & The Node microservices., provided by Kubernetes.

Additionally, to generate replica's with Kubernates HPA autoscaling, there is a .sh script in the Kubernetes-K8s-YAML folder called 'autoscale.sh' which will generate up to 10 replica's of each unit (which scales once 50% CPU load is reached, and then scales back down)

Once autoscaling is enabled, You can generate load by:

1. run ' kubectl run -it --rm load-generator --image=busybox /bin/sh'
2. Hit enter for command prompt
3. run: 'while true; do wget -q -O- http://localhost/app; done'

Development process

The project was built using first, Developing API & mongo on Localhost with Windows 10 and Windows Subsystem for Linux (20.04 Ubuntu).

This was done via a simple Node + Express + MongoDB Install and getting the API to work.

- Next I containerized the app locally in Docker.
- Next I split the app up Using Docker compose and environmental variables.
- Next I created a Dokerfile of the node API container and created an image which was pushed to Docker Hub under ameeno/node-Kubernetes.
- Next I used Kompose convert to create Kubernetes Configurations, which was tested on a local Kubernetes Cluster.

There is load balancing and scalability on both the API and the Database, however as the Database uses persistent storage, it could benefit from Queuing to avoid Database Locking.

Deployment Process Simplified.

There is a Kubernetes-k8s-YAML folder which contains the K8S configuration. There is a script included to create the cluster in any location called "kubectl.sh"

The Repository can be cloned in any location and running kubectl.sh with the included YAML files in the project directory will launch the cluster.

The relied upon docker images are in docker hub as public.

Diagrams & Requirements.

To Develop an API Which add's persistent timestamps to the database, Auto scales, can be deployed automatically.

It must be accessible on http://<IP>/app and accept Post requests to add timestamps to the database.

Get Requests will output contents of Database.

Must auto scale.

Diagram One: Default Namespace when deployed.

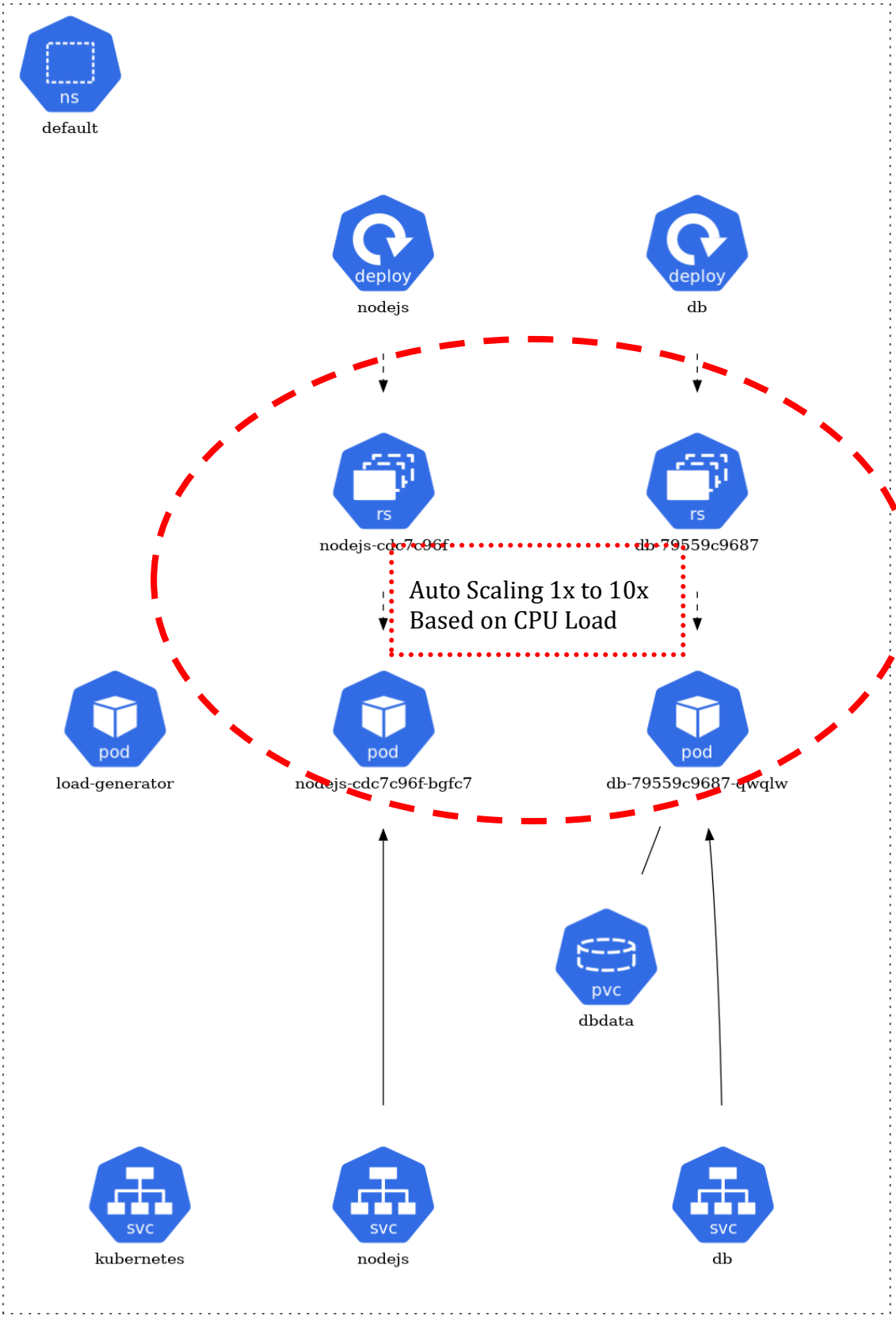
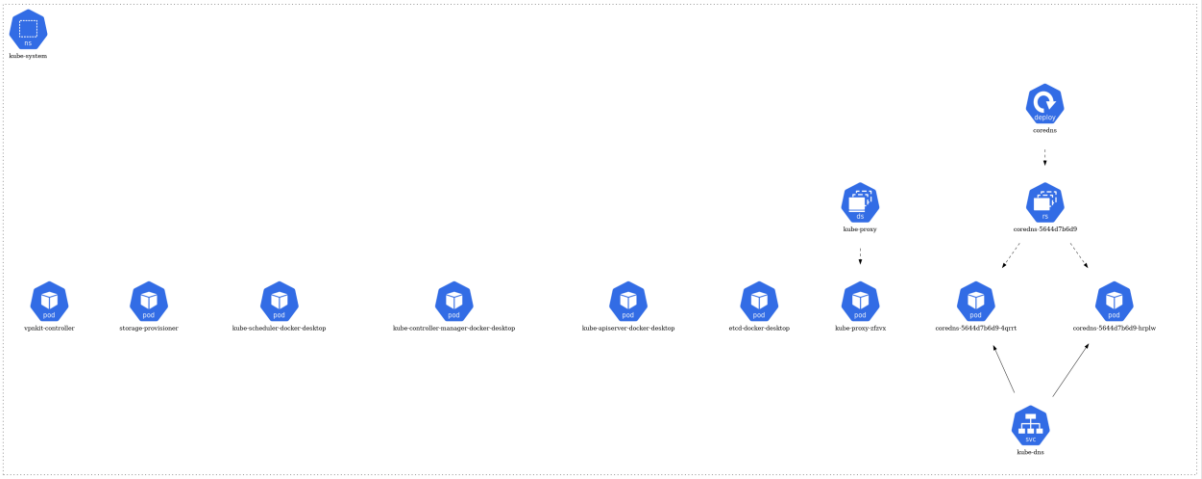


Diagram of Kubernetes System Deployment.



This contains the DNS and resolving namespaces and dns lookups

Diagram of Docker Deployment.: - This is diagram for localhost docker development.