| | |
|---|---|
| online | https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster<br><br>https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/<br><br>`minikube addons enable dashboard`<br><br>Make the Kubernetes Dashboard available by deploying the following YAML definition. This should only be used on Katacoda.<br><br>`kubectl apply -f /opt/kubernetes-dashboard.yaml`<br><br>The Kubernetes dashboard allows you to view your applications in a UI. In this deployment, the dashboard has been made available on port *30000*.<br><br>Click + near terminal and choose<br>View HTTP port 30000 on Host1 |
| Install | Windows<br>VirtualBox, Hyper-V<br><br>Open cmd, ensure that you are using an ***administrative shell***<br><br>Run the following command:<br><br>> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"<br><br>**`> choco install minikube kubernetes-cli`**<br><br>>minikube start |
| Tools | Minikube, Dockerclient, Kops, Kubeadm |

| | |
|---|---|
| Minikube | minikube delete<br>minikube start<br>minikube version<br>minikube dashboard<br>minikube status<br>minikube ip<br>minikube addons list<br>minikube service {service} --url |
| Kubectl | kubectl api-versions<br>kubectl cluster-info<br>https://kubernetes.io/docs/reference/kubectl/cheatsheet/ |

| | |
|---|---|
| | The need for deployments comes more so out of the transition from Replication Controllers to Replica Sets than the lack of capability provided from Replica sets. For this to make sense, one should understand that Replication Controllers were the original resource used for scaling Kubernetes Pods.<br><br>One of the key features of Replication Controllers included the “rolling-update” functionality. This feature allows the pods being managed by Replication controllers to be updated with a minimal/none outage of the service those pods are providing. To accomplish this the instances of the old pods are updated one by one.<br><br>However, Replication controllers were criticized for being imperative and lacking flexibility. As a solution, Replica Sets and Deployments were introduced as a replacement for Replication controllers.<br><br>While Replica Sets still have the capability to manage pods, and scale instances of certain pod, they don’t have the ability to perform a rolling update and some other features. Instead, this functionality is managed by a Deployment, which is the resource that a user using Kubernetes today would mostly likely interact with. |

| | |
|---|---|
| Pods vs Containers | https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16 |
| Deployment | # replication controller<br>#replication set<br># deployment<br>https://www.bmc.com/blogs/kubernetes-deployment/<br>kubectl create -f deployment helloworld.yml<br>kubectl get deployments  or kubectl get deploy<br>kubectl get service,pvc,deployment,pods<br>kubectl get all \| grep hello-world<br>kubectl get rs  or  kubectl get replicaset<br>kubectl rollout status deployment helloworld-deployment<br>kubectl rollout history deployment<br><br>kubectl set image deployment helloworld-deployment k8s-demo=wardviaene/k8s-demo:2<br><br>kubectl rollout status deployment helloworld-deployment<br>curl http://192.168.99.103:30353<br>kubectl rollout history deployment<br><br>kubectl scale --replicas=4 deployment helloworld-deployment<br>kubectl rollout undo deployment helloworld-deployment<br><br>kubectl rollout undo deployment helloworld-deployment —to-revision=3<br>**kubectl delete deployment hello-world**<br><br>ReplicaSet is the next-generation Replication Controller. we recommend using Deployments instead of directly using ReplicaSets, unless you require custom update orchestration or don’t require updates at all.This actually means that you may never need to manipulate ReplicaSet objects: use a Deployment instead, and define your application in the spec section.<br><br>type:deployment vs type:pod<br>Both Pod and Deployment are full-fledged objects in the Kubernetes API. Deployment manages creating Pods by means of ReplicaSets.<br><br>kubectl edit deployment helloworld-deployment<br>spec:<br>  progressDeadlineSeconds: 600<br>  replicas: 4<br>  revisionHistoryLimit: 10 |

| | |
|---|---|
| Pod | kubectl get pods —show-labels<br>kubectl describe pod myPodName -n myNameSpace |

| | |
|---|---|
| Service | kubectl expose deployment helloworld-deployment --type=NodePort --name=example-service<br><br>This is equivalent to **kubectl create -f** the following yaml:<br>apiVersion: v1<br>kind: Service<br>metadata:<br>  name: example-service<br>spec:<br>  ports:<br>  - port: 31001<br>  nodePort: 31001<br>  targetPort: nodejs-port<br>  protocol: TCP<br>  selector:<br>  app: helloworld<br>  type: NodePort<br><br>Kubectl get service<br>kubectl describe service example-service<br>minikube service example-service --url<br>curl http://192.168.99.103:30353<br><br>A Kubernetes Service is an abstraction which defines a logical set of Pods running somewhere in your cluster, that all provide the same functionality. When created, each Service is assigned a unique IP address (also called clusterIP). This address is tied to the lifespan of the Service, and will not change while the Service is alive. Pods can be configured to talk to the Service, and know that communication to the Service will be automatically load-balanced out to some pod that is a member of the Service.<br><br>**ClusterIP:** makes the service only reachable from within the cluster. This is the default ServiceType.<br><br>NodePort :  Kubernetes will allocate a specific port on each Node to that service, and any request to your cluster on that port gets forwarded to the service.<br>Limitation: have to take care of load balancing so that you leverage the power of your pod replicas. if you want to manage load balancing systems manually and set up port mappings yourself.<br><br>LoadBalancer : There needs to be some external load balancer functionality in the cluster, typically implemented by a cloud provider.<br>This is typically heavily dependent on the cloud provider.Every |

| | |
|---|---|
| | Kubectl delete svc example-service |
| LivenessProbe<br>RedinessProbe | liveness probes to know when to restart a Container.<br><br>readiness probes to know when a Container is ready to start accepting traffic. |
| Config | http://pwittrock.github.io/docs/user-guide/configmap/<br>https://medium.com/google-cloud/kubernetes-configmaps-and-secrets-68d061f7ab5b<br><br>**kubectl create -f configmap.yaml**<br>kubectl create configmap language --from-literal=LANGUAGE=English<br>**kubectl get configmap**<br>**kubectl describe configMap test-configmap**<br>**kubectl get configmaps test-configmap -o yaml** |
| Secret | https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/<br><br>kubectl create secret generic apikey --from-literal=API_KEY=123–456<br>kubectl get secret |

| | |
|---|---|
| | Before developers can start using the storage, administrators need to provision persistent volumes. Unlike volumes, persistent volumes are not associated with any specific pod or containers when they are created. They are pre-provisioned storage resources that can be used by developers during the creation of a pod. Once persistence volumes (PersistentVolume) are provisioned by administrators, developers create a claim (PersistentVolumeClaim) to start consuming the storage resources exposed as persistent volumes.<br><br>**Volume**<br>When running multiple containers together in a Pod it is often necessary to share files between those containers; volumes are used for this purpose.<br><br>PERSISTENT VOLUME<br>A PersistentVolume (PV) is a piece of storage in the cluster. It is a resource just like a node is a cluster resource.<br>PVs are volume plugins like volumes but have a lifecycle independent of any individual pod that uses the PV.<br><br>Volumes will be deleted when Pods are deleted, while Persistent Volumes are different entities, completely decoupled from the Pod. PVs are managed by a different set of APIs<br>\ kubectl than Pods and have their own Lifecycle.<br><br>PERSISTENT VOLUME CLAIMS<br>A PersistentVolumeClaim (PVC) is a request for storage. It is similar to a pod. While Pods consume node resources, PVCs consume Persistent Volume resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., can be mounted once read/write or read-only). |

| | |
|---|---|
| Volume | **minikube ssh**<br><br>Sql<br>https://kubernetes.io/docs/tasks/run-application/run-single-instance-stateful-application/<br>https://dev.mysql.com/doc/mysql-getting-started/en/<br><br>if you get this error<br>Can't create database 'database'. (errno: 13)'<br>http://www.webhostingtalk.com/showthread.php?t=361700<br><br>change to the directory above the mysql dir, probably var/lib/:<br>**# cd /var/lib**<br><br>Check the owner and group with<br>**# ls -ld mysql**<br><br>**# chown -R mysql:mysql mysql**<br><br>Nginx<br><br>https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/<br><br>Wordpress<br><br>https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/<br>https://dotlayer.com/how-to-run-wordpress-in-a-kubernetes-cluster/ |

| | |
|---|---|
| Ingress | it sits in front of multiple services and act as a “smart router” or entrypoint into your cluster. Ingress Controllers can technically be any system capable of reverse proxying, but the most common is Nginx.  It is a completely independent resource to your service. You declare, create and destroy it separately to your services.<br><br>https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-ingress-guide-nginx-example.html<br>https://velotio.com/blog/2017/7/5/http-load-balancing-in-kubernetes-with-ingress<br>https://medium.com/@awkwardferny/getting-started-with-kubernetes-ingress-nginx-on-minikube-d75e58f52b6c<br><br>minikube addons enable ingress<br># wait a min for the pod to be up and running<br>$ kubectl get pods -n kube-system \| grep nginx-ingress-controller<br><br>Kubernetes ingress resource lacks of quite important functionality, mainly mutual TLS for incoming connections (currently only support server certificates), supporting SNI (releated with the previous one), non-HTTP traffic load balance, and if you want to use Istio features like Traffic splitting, fault injection, mirroring, header match, etc are not available between ingress controller and backend... I was looking for for an implementation like this, let see if "gateway" finally works and performance is as good as ingress controller.<br><br>Case study:<br>https://engineering.shopify.com/blogs/engineering/iterating-towards-more-stable-ingress |

| | |
|---|---|
| Service discovery | Kubernetes DNS schedules a DNS Pod and Service on the cluster, and configures the kubelets to tell individual containers to use the DNS Service's IP to resolve DNS names.<br>**kubectl get services kube-dns --namespace=kube-system**<br><br>https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough<br><br>You can expose your application via a **ClusterIP** service.**ClusterIP** is the default **ServiceType** and it creates a single IP address that can be used to access its **Pods** which can only be accessed from inside the cluster. If **KubeDNS** is enabled it will also get a series of DNS records assigned to it include an A record to match its IP. This is very useful for exposing microservices running inside the same Kubernetes cluster to each other.<br><br>kubectl run -i --tty busybox --image=busybox --restart=Never -- sh<br>#nslookup azure-vote-back<br><br>Kubectl get pods<br>kubectl exec -it azure-vote-front-fc749d98c-h642z -- /bin/bash<br>#ping azure-vote-back<br><br>Note: to speak between 2 containers in the same pod you don't need dns. Use localhost:port |
| kubeless | https://kubeless.io/docs/quick-start/<br>https://docs.bitnami.com/kubernetes/how-to/get-started-serverless-computing-kubeless/<br>https://github.com/kubeless/kubeless/releases |
| Debugging | https://kubernetes.io/docs/tasks/debug-application-cluster/debug-application/<br>https://sysdig.com/blog/debug-kubernetes-crashloopbackoff/<br><br>Kubectl describe pod {name}<br>kubectl logs {pod} |

| | |
|---|---|
| Auto scale | **Horizontal Pod Autoscaler** watches and scales a pod in the deployment.<br><br>https://www.tutorialspoint.com/kubernetes/kubernetes_autoscaling.htm<br><br>kubectl autoscale deployment azure-vote-front --cpu-percent 50 --min 1 --max 10<br><br>kubectl get hpa<br><br>while true; do curl http://192.168.99.105:30417; done |
| Package manager | https://medium.com/ingeniouslysimple/deploying-kubernetes-applications-with-helm-81c9c931f9d3<br><br>`sudo helm init`<br>helm create mychart<br><br>helm package ./mychart<br><br>helm install --name example3 mychart-0.1.0.tgz<br><br>A **Chart** is a Helm package.<br><br>helm repo update<br>helm install stable/mysql<br>helm inspect stable/mysql<br>helm ls<br>Helm search<br>helm delete wintering-rodent<br><br>https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart/ |

| | |
|---|---|
| Monitoring | minikube addons enable heapster<br>kubectl top node<br>kubectl top pod<br><br>kubectl get pods --all-namespaces<br><br>minikube addons open heapster<br><br>#liveness probe<br>#readiness probe |
| Prometheus | Prometheus is an open source toolkit to monitor and alert<br><br>helm init<br><br>helm --tiller-namespace tiller  version<br>helm install stable/prometheus-operator --name prometheus-operator --namespace monitoring<br><br>kubectl get pods  -n monitoring<br><br>Forward the Prometheus server to your machine so you can take a better look at the dashboard by opening http://localhost:9090<br><br>kubectl port-forward -n monitoring prometheus-prometheus-operator-prometheus-0 9090 http://localhost:9090/graph<br><br>Prometheus has an expression browser for debugging purpose. To have a good-looking dashboard, use Grafana, it has a datasource ready to query on Prometheus.<br><br>kubectl port-forward $(kubectl get  pods --selector=app=grafana -n  monitoring --output=jsonpath="{.items..metadata.name}") -n monitoring  3000<br><br>*The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibition of alerts.*<br><br>kubectl port-forward -n monitoring alertmanager-prometheus-operator-alertmanager-0 9093 |

| | |
|---|---|
| Istio | A service mesh is a network of microservices that makeup applications and the interactions between them.<br><br>https://istio.io/docs/examples/bookinfo/<br>https://github.com/istio/istio/tree/master/samples/bookinfo<br><br>install wnidows<br>———————<br>https://technology.amis.nl/2017/10/25/first-steps-with-istio-on-kubernetes-on-minikube-on-windows-10/<br><br>Install mac/unix<br>———————-<br>curl -L https://git.io/getLatestIstio \| sh -<br>cd istio-0.5.1<br>Open the file istio-1.0.2/install/kubernetes/istio-demo.yaml, search for LoadBalancer and replace it with NodePort.<br>kubectl apply -f install/kubernetes/helm/istio/templates/crds.yaml<br>kubectl apply -f install/kubernetes/istio-demo.yaml<br>kubectl get service -n istio-system<br>kubectl get pods -n istio-system<br><br>https://thenewstack.io/tutorial-blue-green-deployments-with-kubernetes-and-istio/<br>https://medium.com/@bufferings/istio-bookinfo-on-minikube-1-deploy-app-7c953d18913d<br><br>git clone https://github.com/redhat-developer-demos/istio-tutorial.git<br>cd istio-tutorial<br>git checkout book<br><br>Netflix has recently announced it has stopped development of the Hystrix library in favor of the less well-known Resilience4J project.<br><br>https://github.com/IBM/resilient-java-microservices-with-istio<br><br>Azure<br>https://blog.shanelee.name/2018/08/12/istio-on-azure-aks/<br>https://istio.io/docs/tasks/telemetry/using-istio-dashboard/<br><br>Canary release<br>https://docs.microsoft.com/en-us/azure/aks/istio-scenario-routing<br><br>Circuit breaker<br>https://istio.io/docs/tasks/traffic-management/circuit-breaking/ |

| | |
|---|---|
| Patterns | Anti patterns<br>https://github.com/gravitational/workshop/blob/master/k8sprod.md<br><br>Side car<br>https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar<br><br>Ambassador<br>https://docs.microsoft.com/en-us/azure/architecture/patterns/ambassador<br><br>Patterns<br>https://techbeacon.com/7-container-design-patterns-you-need-know<br><br>• One or more containers running within a pod for enhancing the main container functionality (logger container, git synchronizer container); These are **sidecar** container<br>• One or more containers running within a pod for accessing external applications/servers (Redis cluster, memcache cluster); These are called **ambassador** container<br>• One or more containers running within a pod to allow access to application running within the container (Monitoring container); These are called as **adapter** containers |
| RBAC | https://docs.bitnami.com/kubernetes/how-to/configure-rbac-in-your-kubernetes-cluster/<br><br>Minikube<br>`/var/lib/localkube/certs/ca.crt` |
| Kubeadm, Kops | • Kubeadm is in the middle of the stack and it runs on each node, and basically creates and then talks to the Kubernetes API.<br>• Kops on the other hand is responsible for the entire lifecycle of the cluster, from infrastructure provisioning to upgrading to deleting, and it knows about everything: nodes, masters, load balancers, cloud providers, monitoring, networking, logging etc. |

| | |
|---|---|
| Networking | https://www.digitalocean.com/community/tutorials/kubernetes-networking-under-the-hood<br><br>https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/<br><br>https://medium.com/google-cloud/understanding-kubernetes-networking-services-f0cb48e4cc82 |