Storage in Kubernetes

- Storage and stateful workloads are harder in all systems
- Containers make it both harder and easier than before
- StatefulSets is a new resource type, making Pods more sticky
- Bret's recommendation: avoid stateful workloads for first few deployments until you're good at the basics
  - Use db-as-a-service whenever you can

Volumes in Kubernetes

- Creating and connecting Volumes: 2 types
- **Volumes**
  - Tied to lifecycle of a Pod
  - All containers in a single Pod can share them
- **PersistentVolumes**
  - Created at the cluster level, outlives a Pod
  - Separates storage config from Pod using it
  - Multiple Pods can share them
- CSI plugins are the new way to connect to storage

Ingress

- None of our Service types work at OSI Layer 7 (HTTP)
- How do we route outside connections based on hostname or URL?
- Ingress Controllers (optional) do this with 3rd party proxies
- Nginx is popular, but Traefik, HAProxy, F5, Envoy, Istio, etc.
- Note this is still beta (in 1.15) and becoming popular
- Implementation is specific to Controller chosen

CRD's and The Operator Pattern

- You can add 3rd party Resources and Controllers
- This extends Kubernetes API and CLI
- A pattern is starting to emerge of using these together
- Operator: automate deployment and management of complex apps
- e.g. Databases, monitoring tools, backups, and custom ingresses

Higher Deployment Abstractions

- All our kubectl commands just talk to the Kubernetes API
- Kubernetes has limited built-in templating, versioning, tracking, and management of your apps
- There are now over 60 3rd party tools to do that, but many are defunct
- **Helm** is the most popular
- "**Compose on Kubernetes**" comes with Docker Desktop
- Remember these are optional, and your distro may have a preference
- Most distros support **Helm**

Templating YAML

- Many of the deployment tools have templating options
- You'll need a solution as the number of environments/apps grow
- **Helm** was the first "winner" in this space, but can be complex
- Official **Kustomize** feature works out-of-the-box (as of 1.14)
- docker app and compose-on-kubernetes are Docker's way

Kubernetes Dashboard

- Default GUI for "upstream" Kubernetes
  - github.com/kubernetes/dashboard
- Some distributions have their own GUI (Rancher, Docker Ent, OpenShift)
- Clouds don't have it by default
- Let's you view resources and upload YAML
- Safety first!

Kubectl Namespaces and Context

- Namespaces limit scope, aka "virtual clusters"
- Not related to Docker/Linux namespaces
- Won't need them in small clusters
- There are some built-in, to hide system stuff from kubectl "users"

  > kubectl get namespaces

  > kubectl get all --all-namespaces
- Context changes kubectl cluster and namespace
- See ~/.kube/config file

>kubectl config get-contexts

>kubectl config set*

Future of Kubernetes

- More focus on stability and security
  - 1.14, 1.15, largely dull releases (a good thing!)
  - Recent security audit has created backlog
- Clearing away deprecated features like kubectl run generators
- Improving features like server-side dry-run
- More and improved Operators
- Helm 3.0 (easier deployment, chart repos, libs)
- More declarative-style features
- Better Windows Server support
- More edge cases, kubeadm HA clusters

Related Projects

- Kubernetes has become the "differencing and scheduling engine backbone" for so many new projects
- Knative - Serverless workloads on Kubernetes
- k3s - mini, simple Kubernetes
- k3OS - Minimal Linux OS for k3s
- Service Mesh - New layer in distributed app traffic for better control, security, and monitoring