kubectl apply

- Remember the three management approaches?
- Let's skip to full Declarative objects

>kubectl apply -f filename.yml

- Why skip kubectl create, kubectl replace, kubectl edit?
- What I recommend ≠ all that's possible

Using kubectl apply

- create/update resources in a file

  > kubectl apply -f myfile.yaml

- create/update a whole directory of yaml

  > kubectl apply -f myyaml/

- create/update from a URL

  > kubectl apply -f https://bret.run/pod.yml

- Be careful, lets look at it first (browser or curl)

  > curl -L https://bret.run/pod

  - Win PoSH? start https://bret.run/pod.yml

Kubernetes Configuration YAML

- Kubernetes configuration file (YAML or JSON)
- Each file contains one or more manifests
- Each manifest describes an API object (deployment, job, secret)
- Each manifest needs four parts (root key:values in the file)

  apiVersion:

  kind:

  metadata:

  spec:

Building Your YAML Files

- **kind:** We can get a list of resources the cluster supports

  > kubectl api-resources
- Notice some resources have multiple API's (old vs. new)
- **apiVersion:** We can get the API versions the cluster supports

  > kubectl api-versions
- **metadata:** only name is required
- **spec:** Where all the action is at!

Building Your YAML spec

- We can get all the keys each **kind** supports

  > kubectl explain services --recursive

- Oh boy! Let's slow down

  > kubectl explain services.spec

- We can walk through the spec this way

  > kubectl explain services.spec.type

- spec: can have sub spec: of other resources

  > kubectl explain deployment.spec.template.spec.volumes.nfs.server

- We can also use docs
  - kubernetes.io/docs/reference/#api-reference

Dry Runs With Apply YAML

- New stuff, not out of beta yet (1.15)
- dry-run a create (client side only)

  > kubectl apply -f app.yml --dry-run
- dry-run a create/update on server

  > kubectl apply -f app.yml --server-dry-run
- see a diff visually

  > kubectl diff -f app.yml

Labels and Annotations

- **Labels** goes under metadata: in your YAML
- Simple list of key: value for identifying your resource later by selecting, grouping, or filtering for it
- Common examples include tier: frontend, app: api, env: prod, customer: acme.co
- Not meant to hold complex, large, or non-identifying info, which is what **annotations** are for
- filter a get command

  > kubectl get pods -l app=nginx
- apply only matching labels

  > kubectl apply -f myfile.yaml -l app=nginx

Label Selectors

- The "glue" telling Services and Deployments which pods are theirs
- Many resources use Label Selectors to "link" resource dependencies
- You'll see these match up in the Service and Deployment YAML
- Use Labels and Selectors to control which pods go to which nodes
- Taints and Tolerations also control node placement

Cleanup

- Let's remove anything you created in this section

  > kubectl get all

  > kubectl delete <resource type>/<resource name>