WWU MÜNSTER

sciebo

sciebo ng

University of Münster

WWU IT, Röntgenstr. 7-13, 48149 Münster

Sciebo Site Report

- started in 2015 for NRW
- serving branded ownCloud to over 30 academic institutions
- 235k registered users (including recently expired accounts, 6 month time limit) (+40k over last CS3)
- 5k project boxes (transferable group accounts)
- 2.8PB of user data including snapshots (30 days)
- moved about 2/3 of production to kubernetes (k8s) over the last year

Learning about k8s

- tons of resources on how to deploy pre-existing apps to some pre-existing cluster, where you pay for all the nice features with your €€€ and metadata
- often you get stuck in the "why is this not working?" loop because your cluster lacks magic
- not a lot of resources on
  - how to run a cluster
  - how to package applications
  - on good container design (a lot of docker baggage out there)
- very good reference though

Goals for this talk

- establish a basic high level understanding of kubernetes
- present choices made at sciebo in the deployment of k8s
- sketch an ownCloud deployment

Running a k8s cluster

- containers are (usually) just linux processes
- k8s is in some sense init system for your cluster, managing where processes run and giving you a way to interact
- we have to add things to make it a "productive", logging, storage, etc.
- the combination of these choices make for a kubernetes distribution, so we are basically running "k8s from scratch"
- today there are probably more viable distributions, but many are "opinionated open source" which is newspeak for "propietary heaps of dark patterns to make you buy support"

Our choices in running a k8s cluster

- installation of k8s via `kubeadm`
- main cluster storage: rook-ceph for all services (DB servers, Logging servers etc., a good choice, good community support for both rook and ceph and it is cloud native)
- ownCloud user data storage: GPFS via hostPath volumes (for user data, it worked for us in the past, but definitely under scrutiny for future deployments)
- HAProxy + NGinxIngress for ingress networking (see last talk)
- Prometheus, Loki + Grafana for monitoring and logging (good architecture, distinct lack of Java, great community support, made for the job)

Practical notes involved running a k8s cluster

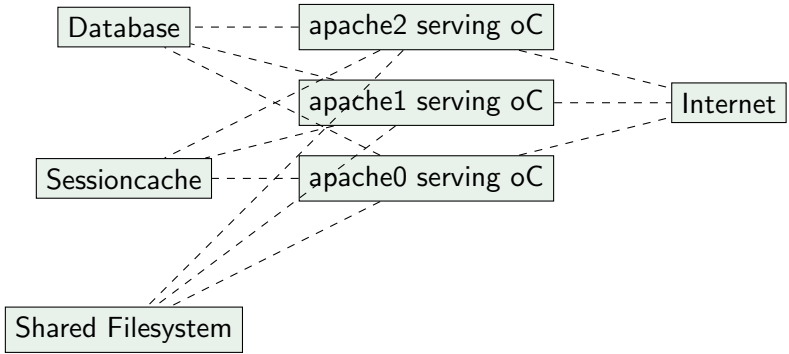
- we use pre-existing and community supported helm charts for pretty much all components
- ususally just some “install.txt” alongside k8s resource manifests and helm value files in git for base services
- whole cluster can be build from git without ever leaving command line
- setting up a k8s cluster and deploying your workload there is a lot less hassle than fiddling around with VMs or physical servers (once you get the hang of it)
- multiy-tenancy is a different story though

Moving applications to k8s

- at first k8s seems overly complex
- what k8s basically does is give you the language to express running applications in the form of a system of API objects, e.g.
  - a linux process is a container and might need configs and volumes, all of which is combined in a Pod
  - Deployments create ReplicaSets which run a number of identical Pods
  - Services provide DNS names for deployments of pods, so you can address them without having to know exactly where they run
  - custom resources like ServiceMonitors can even represent relations such as "Prometheus, monitor this service"
- not k8s is complex, but running applications is

Moving applications to k8s

How an ownCloud deployment sees the world



Moving applications to k8s

- `helm` is usually presented as package manager for k8s, but much more important it is a templating mechanism for kubernetes
- helm utilizes the go templating language and you can do all sorts of things, such as
  - default values: `{{ .Values.optional | default "False"}}`
  - loops and branching
  - nested templating
- nice usage "on the fly" possible: `helm template . | kubectl apply -f -`
- `kustomize` seems at first to be easier to get around with, but actually isn't and most people run into its limitations rather quickly

Moving applications to k8s

- we need to build a container containing all the static parts, i.e. Dockerfile running `yum install httpd php` etc. + ownCloud files + sciebo branding
- helm templates for all sorts of things:
  - ConfigMap containing `config.php` with database, cache, datadir and hostname
  - ConfigMap `php.ini` with sessioncache set to cache server
  - the actual Deployment where everything is put together
  - Ingress ressource so traffic incoming for hostname is routed to our pods
- some CI scripting to push our production ready helm chart to some central repo *sciebo*

Moving applications to k8s

- we end up with a helm chart which can install a production ready ownCloud via
  `helm install -n example example-owncloud sciebo/owncloud --values values.yml`
- `values.yml`:

```
replicas: 3
database: mariadb.example.svc
cache: redis.example.svc
datadir: /gpfs/sciebo/example
hostname: example.sciebo.de
```

- of course some omissions: opcache needs emptyDir, CronJobs, fluentbit sidecar because ownCloud can not log to stdout, workaround because oC/NC needs writable `config.php`...
- the same for redis, mariadb...

Plans for the future

- service discovery across cluster borders for true multi-site deployment and availability and single pane of glass monitoring
- more automation of deployments using argocd and gitlab CI
- a lot of the infrastructure that has grown around our ownCloud deployments, LDAP, OnlyOffice, our own customer account management portal and assorted
  ... procrastination candidates still need to be moved
- updating our host system (no fun due to propietary stuff)

Thanks!

Thanks!