Advanced Kubernetes

Part I - Deployment and deployment strategies

Marius Dieckmann[1,2,3]
Marius.Dieckmann@computational.bio.uni-giessen.de

EMBL-EBI Cloud Consultants[4]
cloud-consultants@ebi.ac.uk

[1] Justus-Liebig-Universität Gießen

[2]NFDI4Biodiversity

[3]de.NBI

[4]EMBL-EBI

February 9, 2021

Preliminary remarks

- The course is designed to be hands-on
- A demo application will be deployed step-by-step
- More advanced techniques will be added
- The focus is to (generally) introduce you concepts and their ideas
- Individual tools implement these concepts and ideas
- These tools are fairly complex, expect things to not work at first try

Program

1. Important basic concepts
   - 1.1 Semantic versioning
   - 1.2 Container building
2. Useful K8s resources
   - 2.1 Secrets
   - 2.2 ConfigMaps
   - 2.3 DownwardAPI
   - 2.4 Labels and Annotations
   - 2.5 Health checks
   - 2.6 Limits and Requests
   - 2.7 StatefulSet
3. Hand-on

Semantic versioning

- Idea: Versions and version bumps have a meaning
- Defined in Backus-Naur Form grammar
- Schema: major.minor.patch - [pre-release/build]
- Often used with leading "v."
- Meaning:
  - major: Breaking api change
  - minor: Major feature update
  - patch: Bugfixes
- Used to understand significance of change and decide which new version to deploy
- Used to disambiguate between different release channels (stable, dev, alpha,...)
- Multiple CI/CD tools available that can handle these versioning schema

Container building

- Various options:
  - Dockerhub
  - Quay.io
  - private registries: e.g. Harbor + CI/CD tool
  - ...
- Conditional execution of branches/releases
- Tagging based on tags/branches
- Tagging strategy can vary, but should be documented
- Run CI first
- Strategy example:
  - Builds on pulls on master and develop with branch tagging
  - Builds on releases/tags with SemVer tags

Useful Kubernetes resources

Configuration options etc...

- ConfigMaps:
  - Used to store configuration of components in files
  - Format independent, yaml or toml are often used
  - No need to package configuration with image, no need to rebuild image on config change
  - Can be mounted as files or environment variables
  - Not viable for secrets (e.g. passwords)
- Secret
  - Similar to ConfigMaps
  - Can be used for secrets like passwords
  - Can be mounted as files or environment variables
- Downward API
  - Volume mount to get information about the deployment
  - Can e.g. be used to get the current container tag which indicates the deployed version

Deployments in Kubernetes

Deployment are the fundamental building blocks of almost all long-running services in Kubernetes like websites or API endpoints

- Definitions:
  - Application: A set of individually deployed components, e.g.: webfrontend, backend, database
  - Service: Kubernetes service
- Label and annotations:
  - Label describe the deployment and can be used in select statements
  - Annotations also describe the deployment, but are not used in select statements, they are often used to apply advanced config options
  - Labels are often composed, e.g. indicating version, application and application component

Deployments - Nice extras

- Deployments use by default a rolling update mechanism; pods are updated one by one, therefor a service should always be available
- Build-in health checks for API endpoints
  - shell cmd or http request
  - liveness probe: checks periodically if a program is still in working state
  - startup probe: waits until pod is ready, e.g. if programs needs to load large file
- Pod is considered "Running" only when health checks succeed
- Horizontal pod autoscaling (HPA) to automatically scale a deployment

Requests and Limit

- Can target any resource (including custom ones)
- Should be attached to every container
- Requests:
  - Used for scheduling
  - K8s will run pods with this request only on nodes with sufficient resource
  - Scheduling will be delayed if insufficient resources are available
- Limit
  - Used to determine maximum resource amount
  - Needs to be $\geq$ Request
  - Going over limit can cause pods to be evicted

Scheduling

- Priority: If higher, lower pods will be evicted
- Non-preempting: No active eviction
- Policies:
  - Scenario 1: Pod within Requests & Limits
    - Only evicted on priority or system pressure
  - Scenario 2: Pod over Requests but within Limits
    - Pod can be evicted if scheduler requires space
  - Scenario 2: Pod over Requests & over Limits
    - Pod will be evicted or throttled (uncompressible vs. compressible)

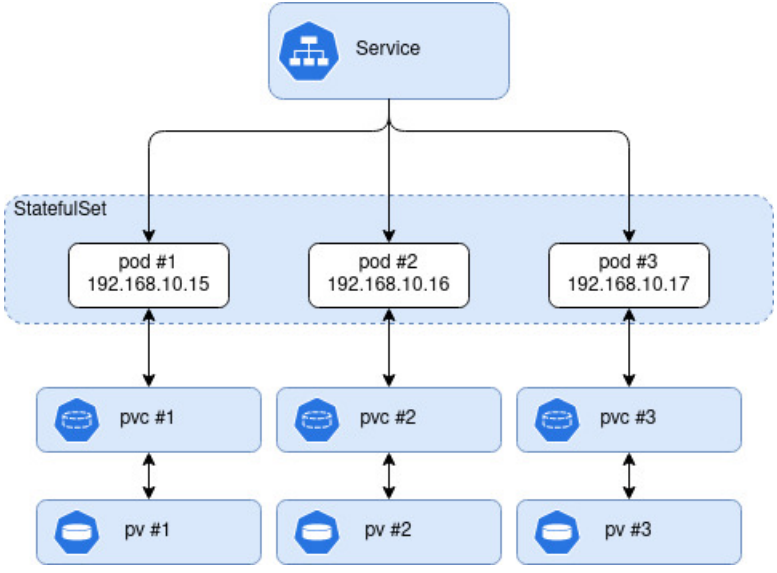
HPA - Horizontal Pod Autoscaling

- Kubernetes can automatically scale deployments
- Controller called Horizontal Pod Autoscaler
- Define min/max number of running pods
- Define target metrics (e.g. CPU, Memory or custom)
- Can be applied to existing Deployment
- Requires Limits and Requests attached
- Requires metrics-server

Excursus: StatefulSet

- Designed for applications that require state
  - fix IP adresses
  - deterministic DNS addresses and pod names
  - multiple single attach PVs based on PV templates
  - Ordered/Graceful deployment/scaling
  - Ordered update rollouts
- Use-cases:
  - Certain distributed systems that require fixed IPs
  - Distributed databases (MongoDB, MySQL, redis)

Excursus: StatefulSet



Hands-on part 1

- Instructions can be found in the following repo:
  `https://github.com/MariusDieckmann/CanaryDemo`