Scaling Up Apache Airflow To Enterprise Level

Martijn Beenker - Data Engineer @ Avanade

Subsurface LIVE

The Cloud Data Lake Conference

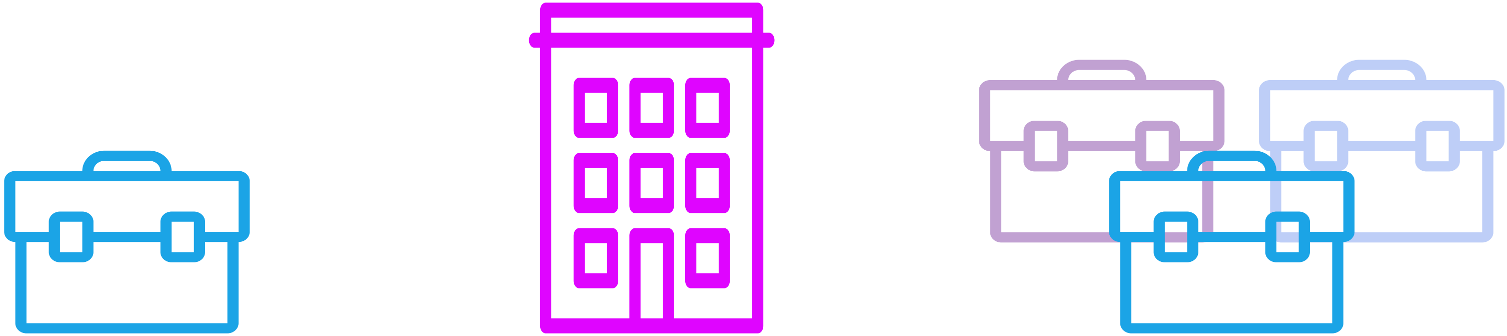
Agenda

1. Quick introduction to containers, k8s and Apache Airflow
2. Why should you care about designing for scale?
3. What went wrong?
    a. Bye bye database: designing for auto-scaling
    b. Overcrowding: database concurrency
    c. No more room at the inn: designing your network carefully
    d. Don’t look back: update your default app settings
    e. Do you copy?: enhancing user proficiency
4. Q & A

Sync Watches on Containers, Kubernetes & Airflow

Sync Watches on Containers, Kubernetes & Airflow

Sync Watches on Containers, Kubernetes & Airflow



Why Should You Care About Designing For Scale?

When growing in # users, Apache Airflow will start to **behave differently over time**. If you stay unprepared, your system will **experience unavailability or performance degradation**.

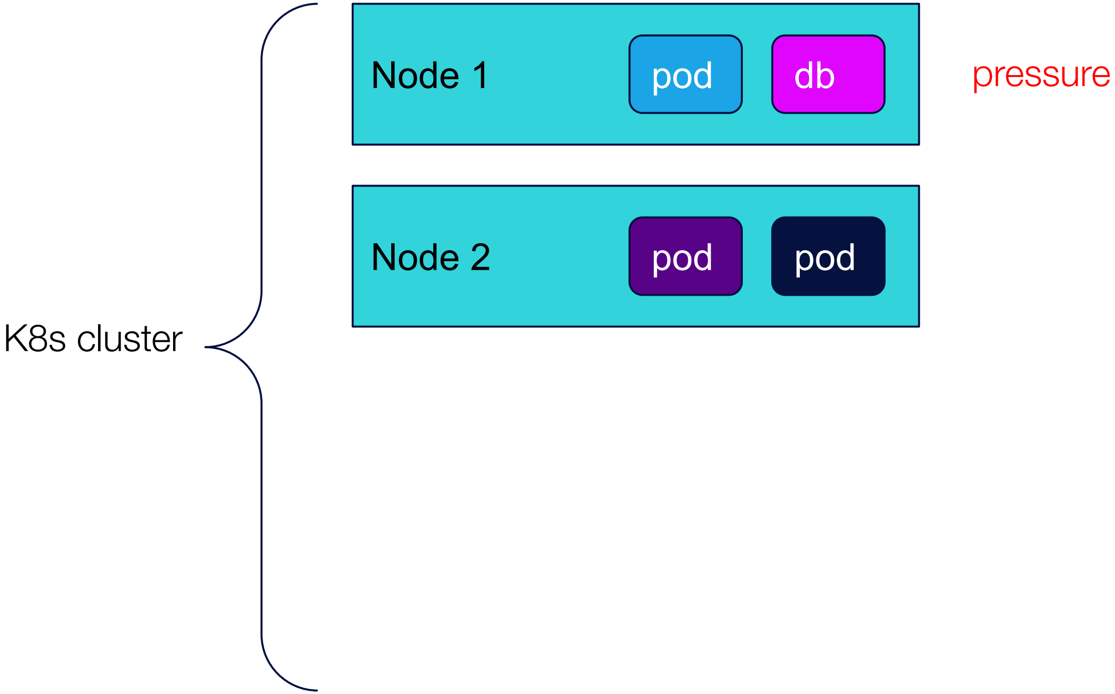
With a big user base you **downtime becomes expensive**. Downtime affects many **downstream systems** (reports, analysis etc) and results in high numbers of **idle engineers**.

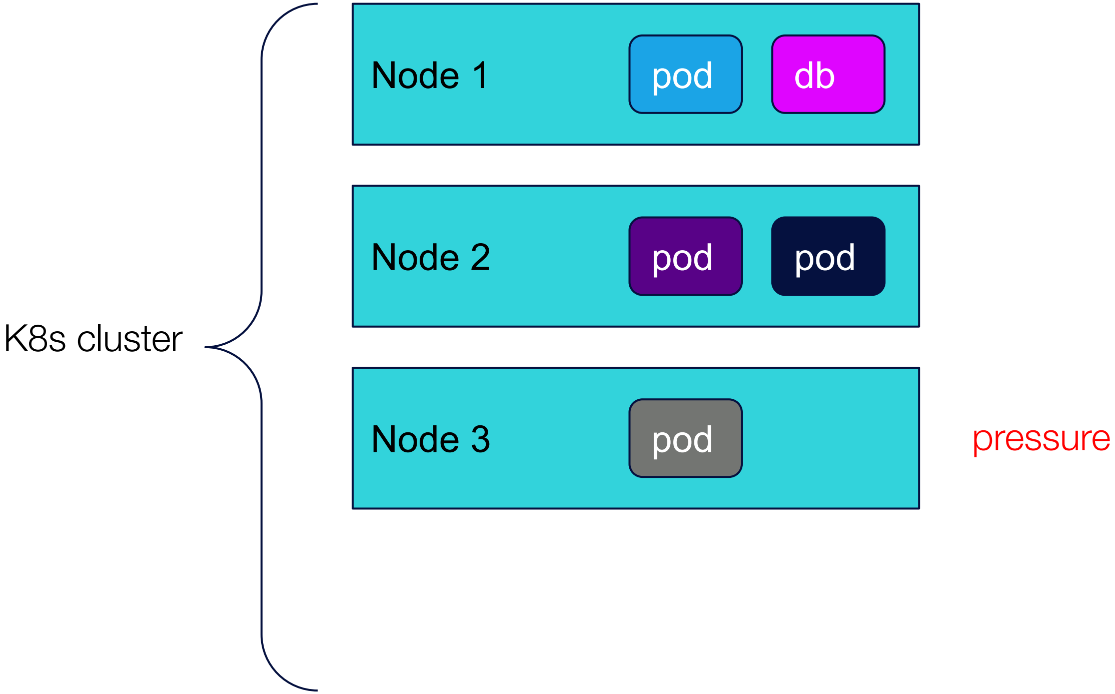
Bye Bye Database

(Failure 1)

In-Cluster DB

Hosting your db backend as pod in the cluster



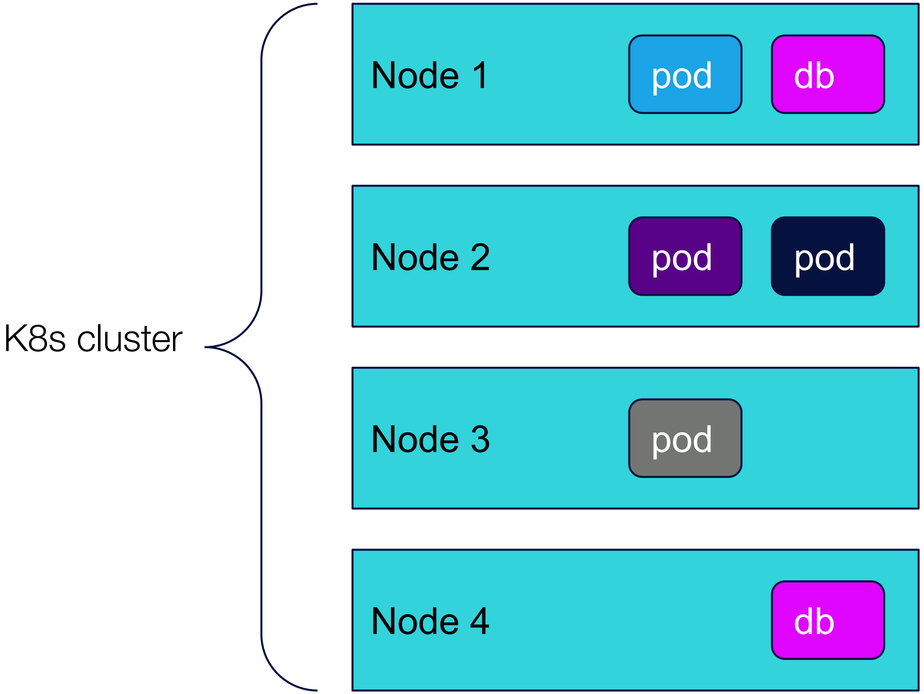


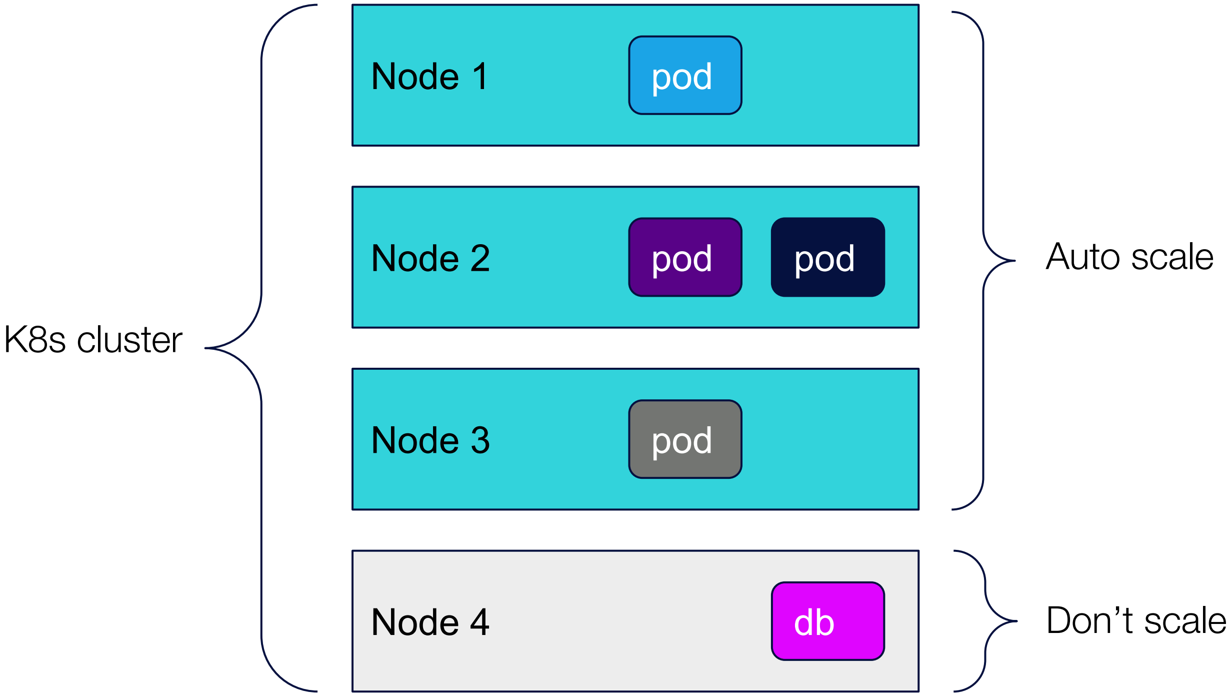
In-Cluster DB

High node utilization will lead to the provisioning of more nodes.

In-Cluster DB

Problem: Database unavailability because of workload redistribution.





Taint the node, add **toleration** and **node affinity** on the db pod

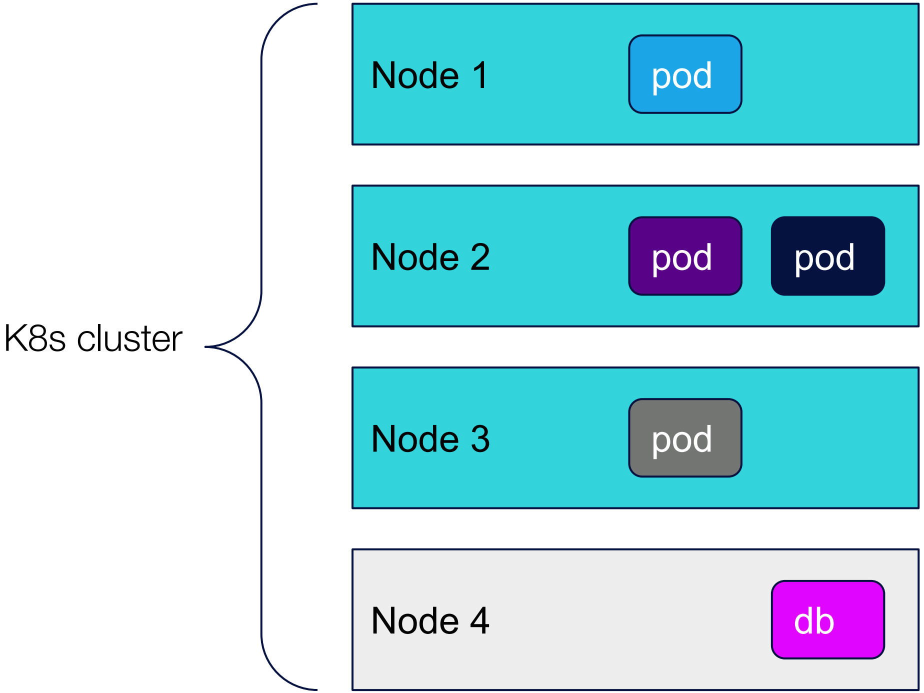
In-Cluster DB

Answer: Don’t auto-scale your database node pool.

- Create different node pools. Host worker pods on a node pool that auto-scales. Host the db pod on a non-scalable node pool.
- Monitor your un-scalable node pools.
- Think about the VM SKU used for each node pool. Our db used a 32 core instance which on a cpu optimized SKU.

Overcrowding The Database

(Failure 2)



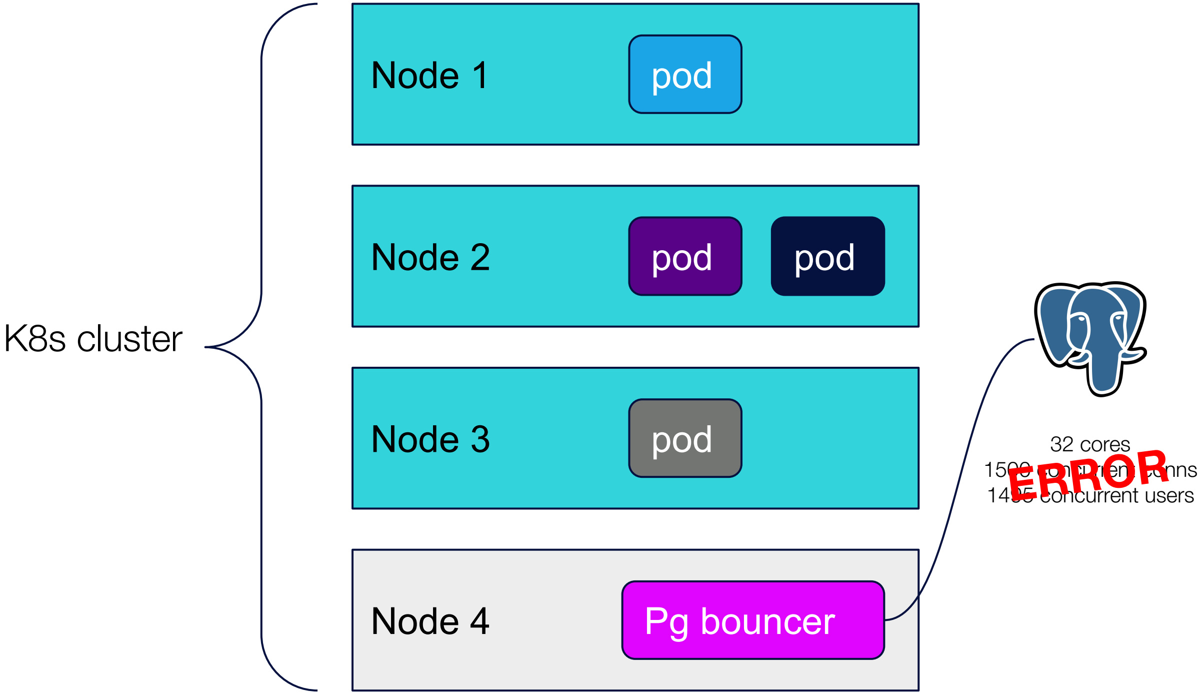
32 cores
1500 concurrent conns
1495 concurrent users

PaaS Database

Problem: In-Cluster DB required to much maintenance

Answer: PaaS DB

- Significantly less maintenance
- Better performance
- Easier to monitor
- Releases are no longer affected by helm chart update



K8s cluster

Node 1

pod

Node 2

pod

pod

Node 3

pod

Node 4

Pg bouncer

32 cores
1500 concurrent conns
1495 concurrent users

ERROR

PaaS Database

Problem: We ran out of concurrent connections

Answer: Connection Pooling

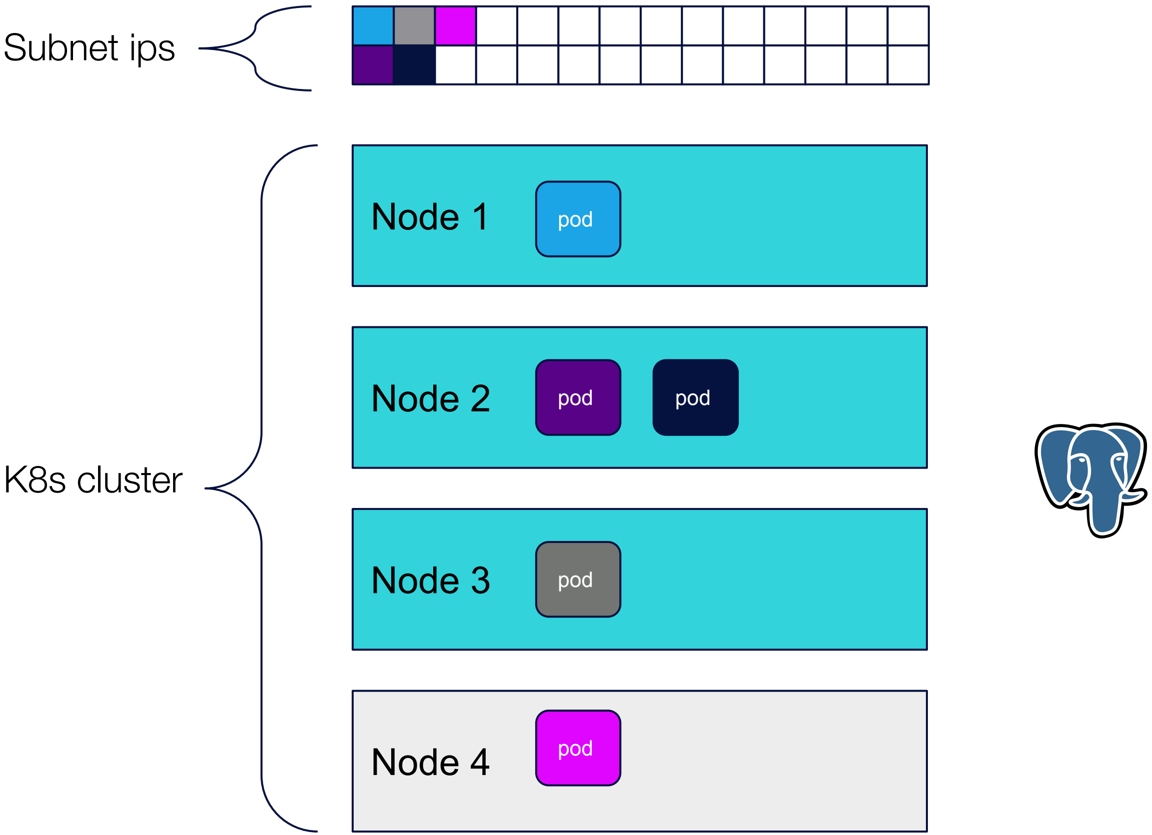
- Default is turned off
- Takes some tinkering, but pays off

Additional Advice

- Don't wait with implementing your connection pooler.
- Azure PostgreSQL Flexible Server includes PgBouncer

No More Room In The Network

(Failure 3)



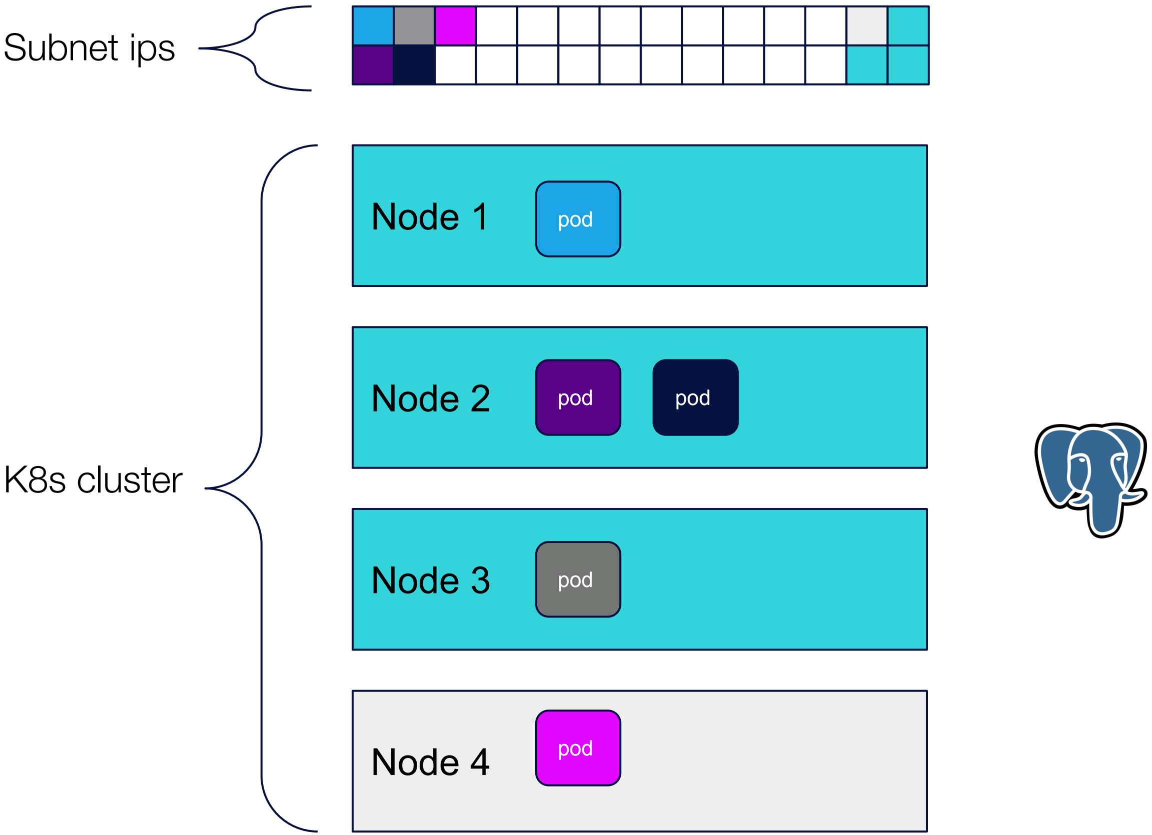
Network Design

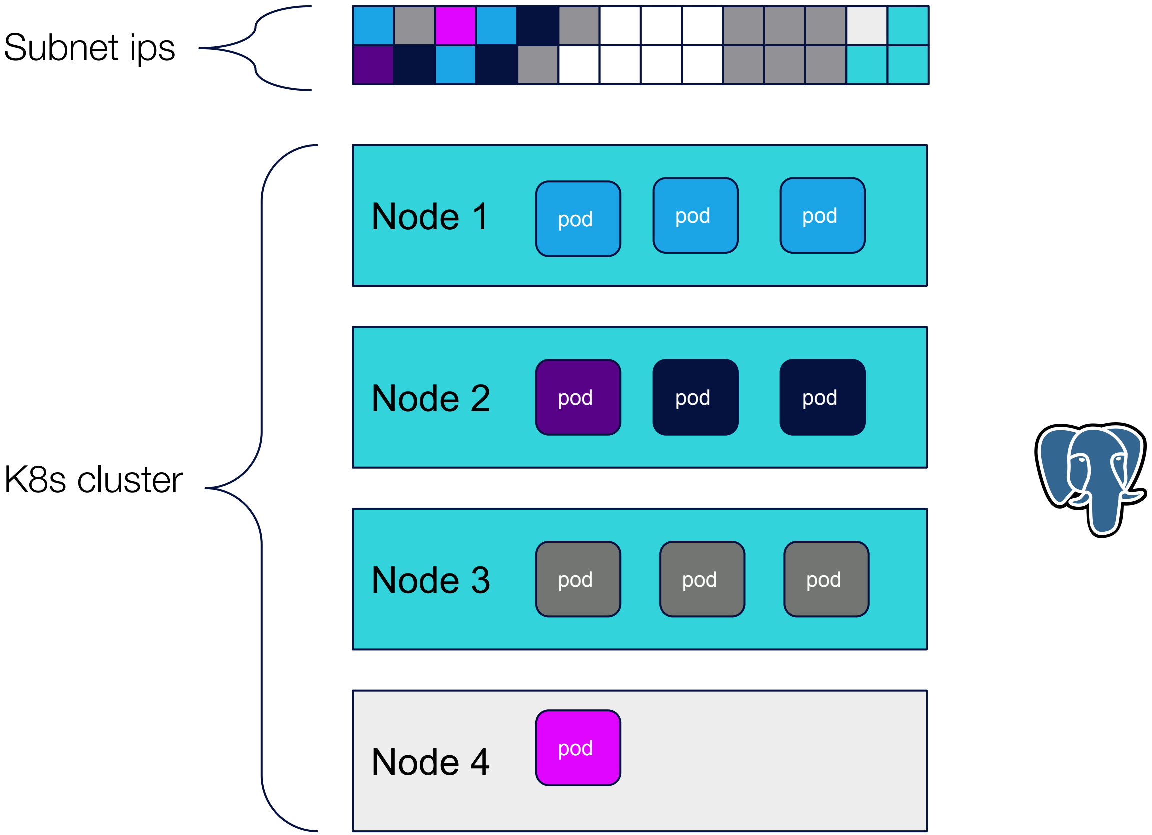
Problem: We ran out of available subnet space

- It's not just your scheduler, triggerer, statsd, web server and worker pods

Network Design

- It's not just your scheduler, triggerer, statsd, webserver and worker pods
- It's also your VM instances



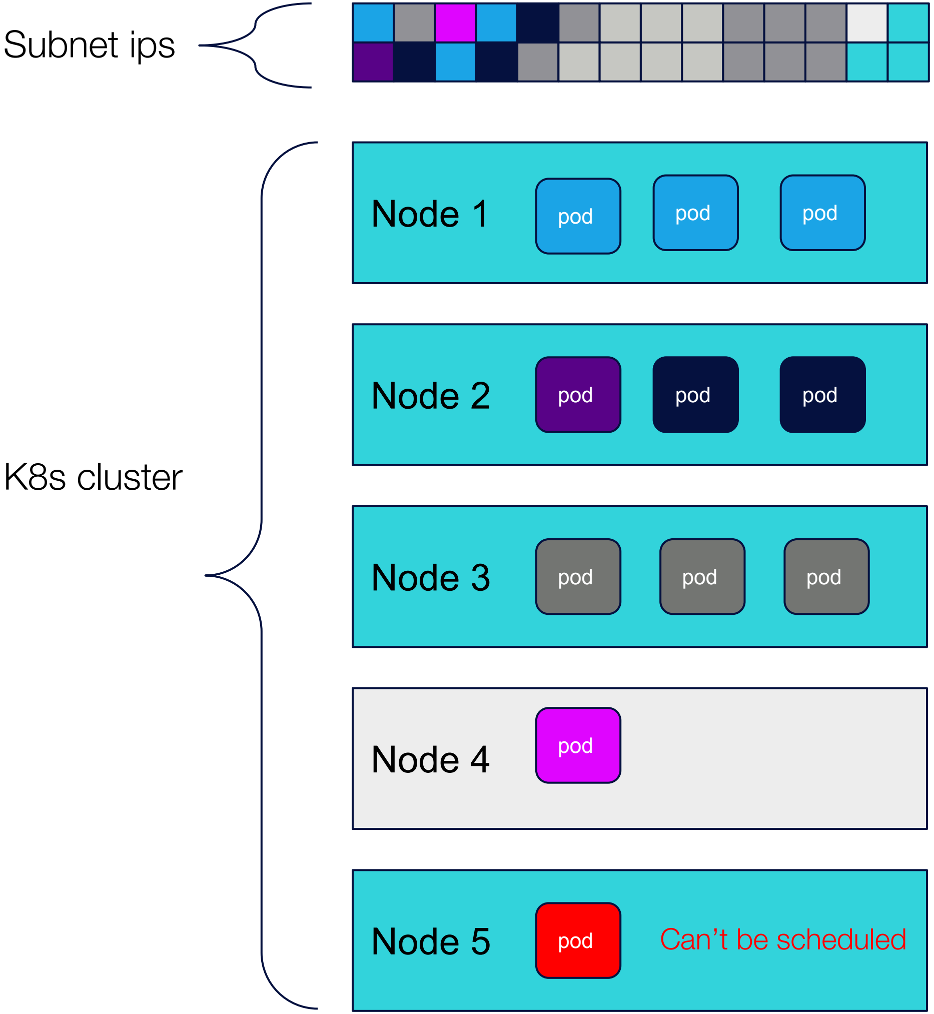


Network Design

- It's not just your scheduler, triggerer, statsd, webserver and worker pods
- It's also your VM instances
- And the pods in kube-system namespace

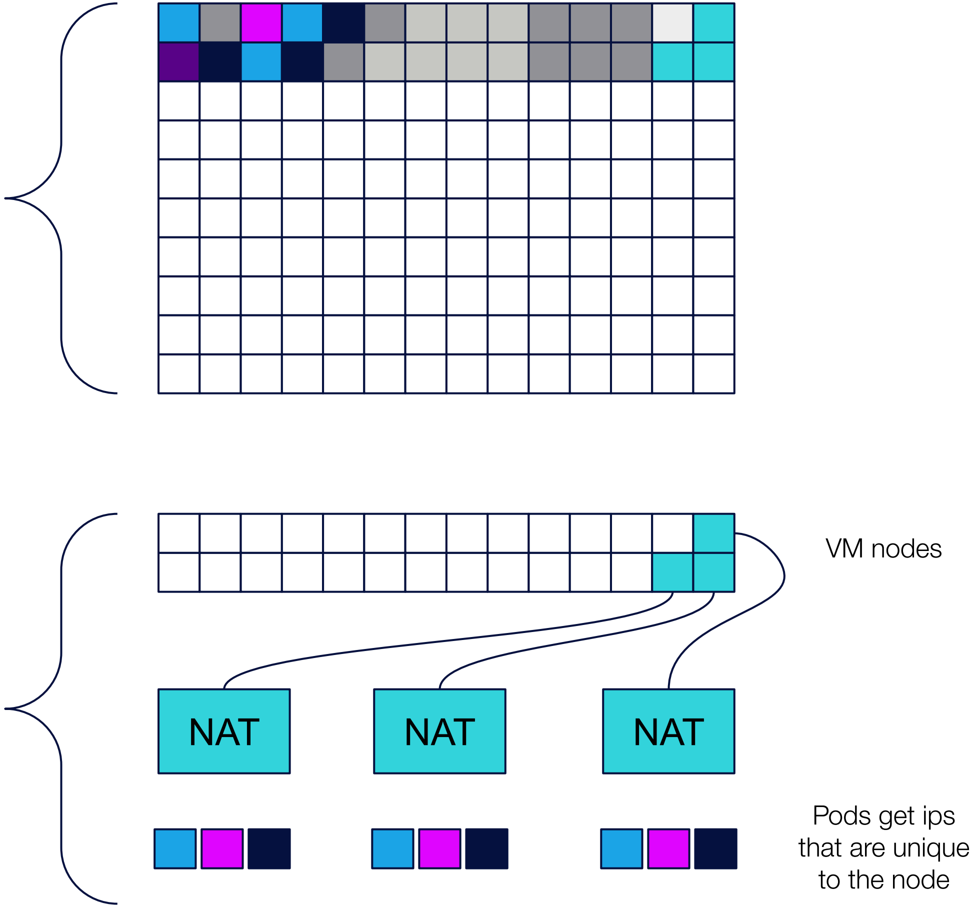
Network Design

Problem: No available space == no new (worker) pods



Azure CNI

(Allow for at least 200.000 hosts)



VM nodes

Pods get ips that are unique to the node

Kubenet

(smaller subnet + NAT)

Network Design

Answer: use large networks or use Kubenet

- Either plan for a very big subnet with Azure CNI
- Or design a network that leverages Kubenet
- Trade-off is scalability vs CPU overhead for NAT

User Proficiency & Trust

(Failure 4)

Support Your Community

Support

Deliver **real-time support**. Especially to your early-adapters.

Incidents

Separate incidents from support. Have a ticket system ready.

Communicate **clearly and early** on any incident to gain trust from your user community.

Community

Let the local community mature and learn to **take care of itself**. Have an internal Stack Overflow ready. Monitor and do not tolerate flaming.

Don’t Review Application Settings

(Failure 5)

Application Settings

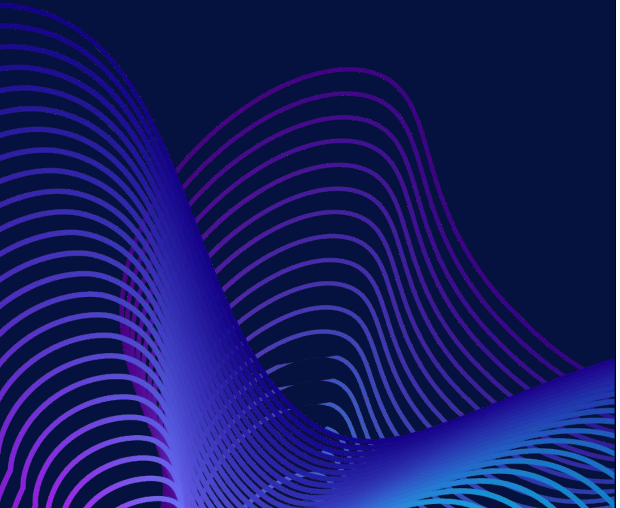
Problem: application settings can cause underutilization of the platform’s resources

Answer: keep updating your settings

- Have an environment where you can simulate production workloads
- Experiment with settings here

Setting	What does it control	Default	Try…
worker_pods_creation_batch_size	Number of Kubernetes Worker Pod creation calls per scheduler loop.	1	20
max_dagruns_to_create_per_loop	Max number of DAGs to create DagRuns for per scheduler loop.	20	50
max_dagruns_per_loop_to_schedule	How many DagRuns should a scheduler examine (and lock) when scheduling and queuing tasks.	20	50
parallelism	This defines the maximum number of task instances that can run concurrently in Airflow regardless of scheduler count and worker count	32	128

Recap



1. Don’t auto-scale your database node/connection pooler node
2. Database Capacity
   a. Use a PaaS database to reduce toil on the dev team
   b. Use a connection pooler, no matter the cluster’s size
3. Plan & design your network to accommodate 200.000 hosts
4. Plan & implement appropriate comms channels
5. Update the default application settings when scaling up

Big Thank You To My Colleagues

Glenn
Tycho
Jaminu
Krijn
& all the other folks at Rabobank

Martijn Beenker

Senior Data Engineer, Senior Data Platform Engineer, Avanade

- Started as Business Intelligence & Data Warehouse Developer.
- Passionate about DevOps practices, borrowing them from the app dev realm, applying them to the data realm
- Believes in automation to enable a more happy work environment and work-life balance
- Likes building strong engineering teams.

Reach out on:

in nl.linkedin.com/in/martijnbeenker
github.com/lowerkees

Subsurface LIVE

The Cloud Data
Lake Conference

Thank You