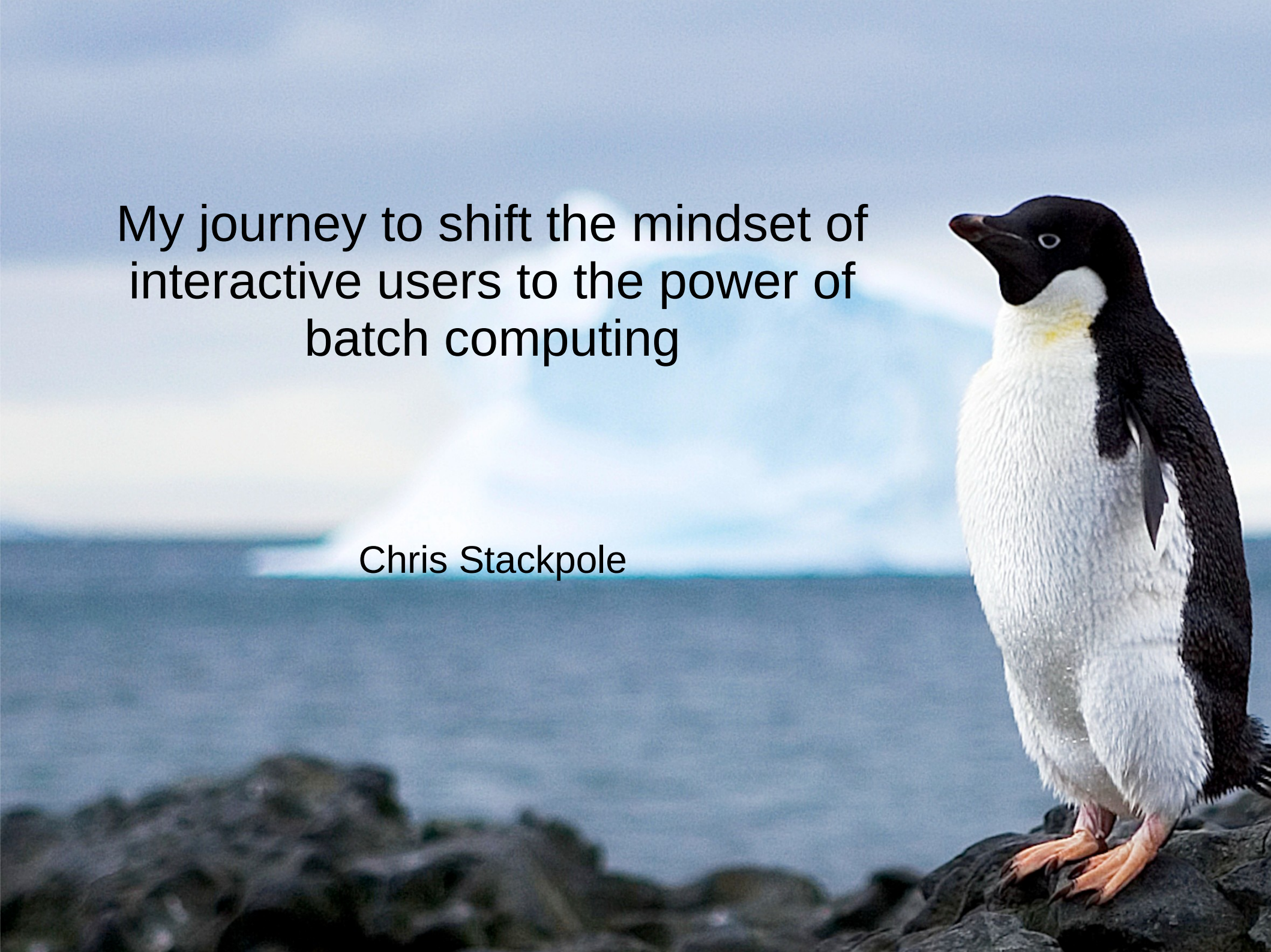
My journey to shift the mindset of interactive users to the power of batch computing

Chris Stackpole

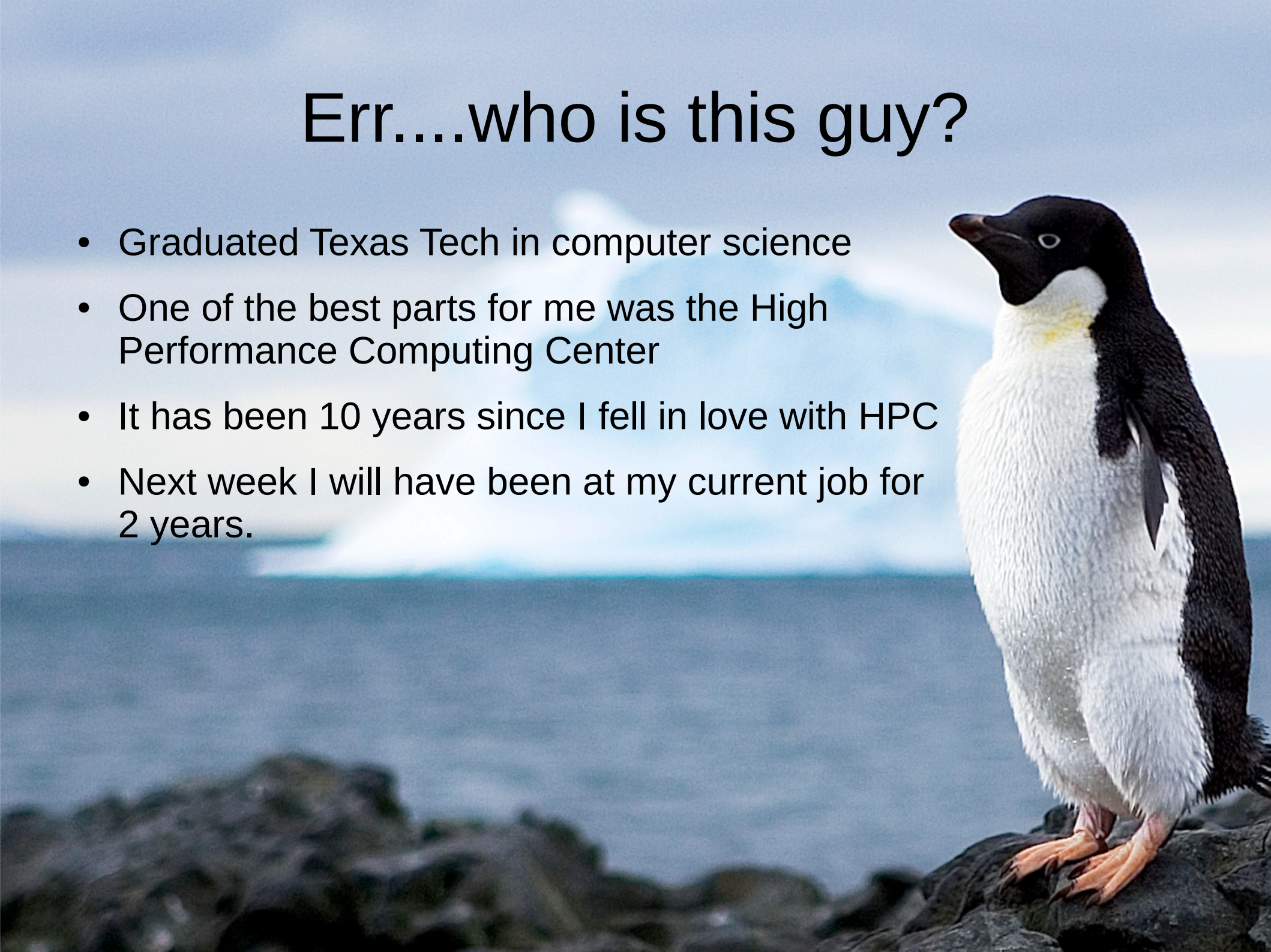



This presentation in no way reflects my employers stance on anything at all...


Background image:
Antarctic Penguin by Christopher Michel
CC BY-NC 2.0
www.flickr.com/photos/cmichel67/83253268

Err....who is this guy?

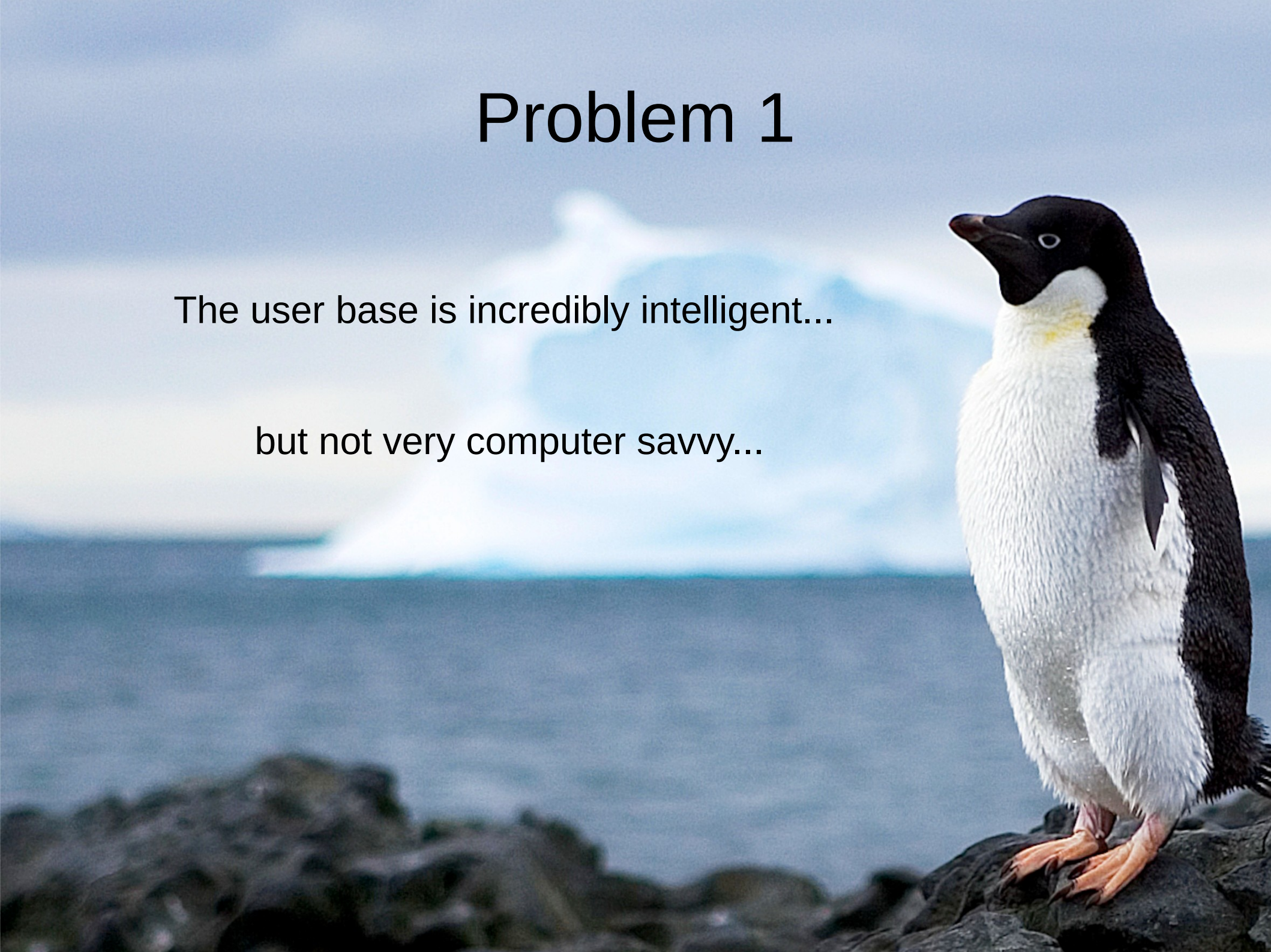
- Graduated Texas Tech in computer science
- One of the best parts for me was the High Performance Computing Center
- It has been 10 years since I fell in love with HPC
- Next week I will have been at my current job for 2 years.



Problem 1

The user base is incredibly intelligent...

but not very computer savvy...



Problem 2

The work flow.

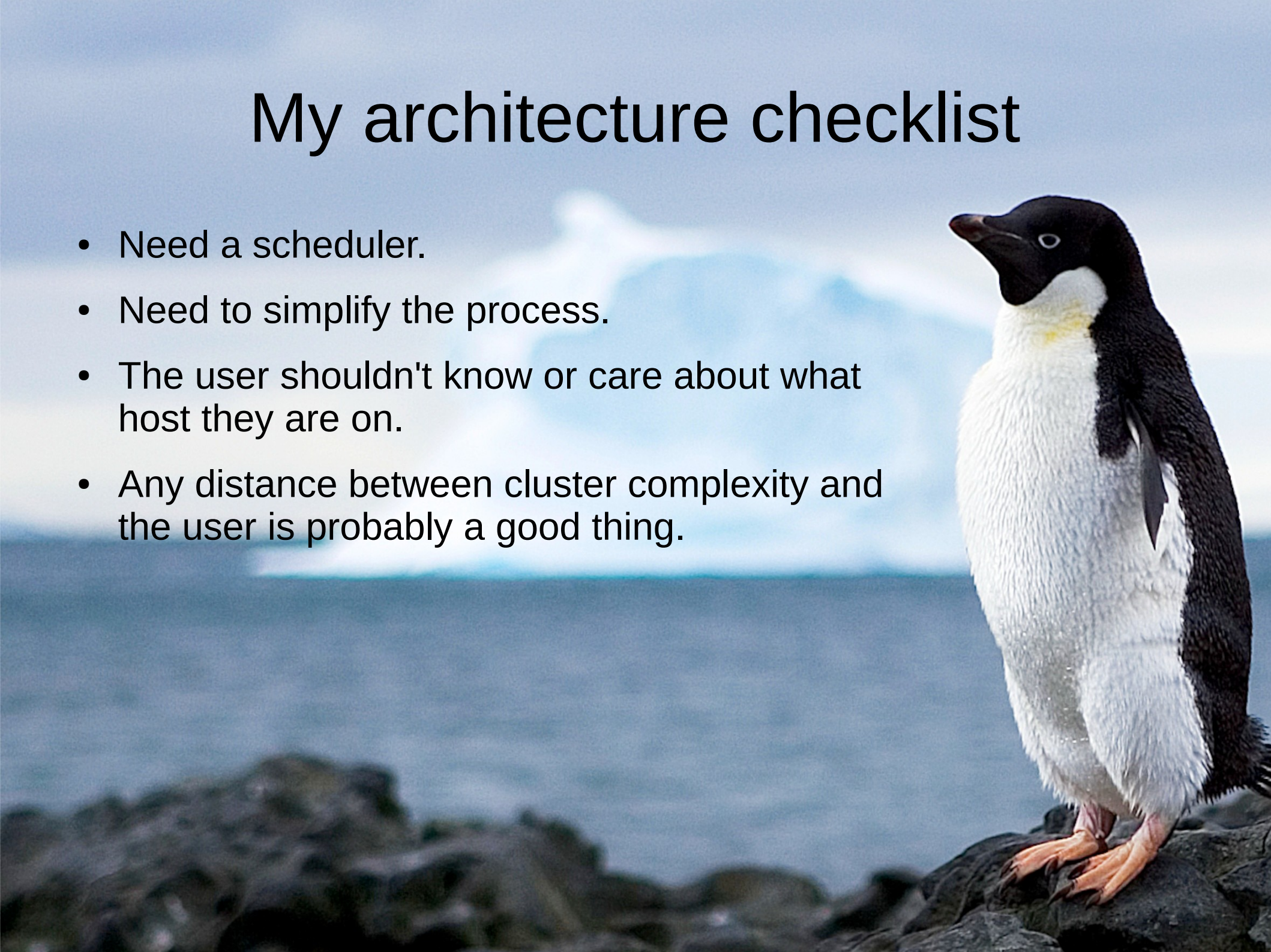
Application on Desktop->Log in node->

Interactive Shell->Application

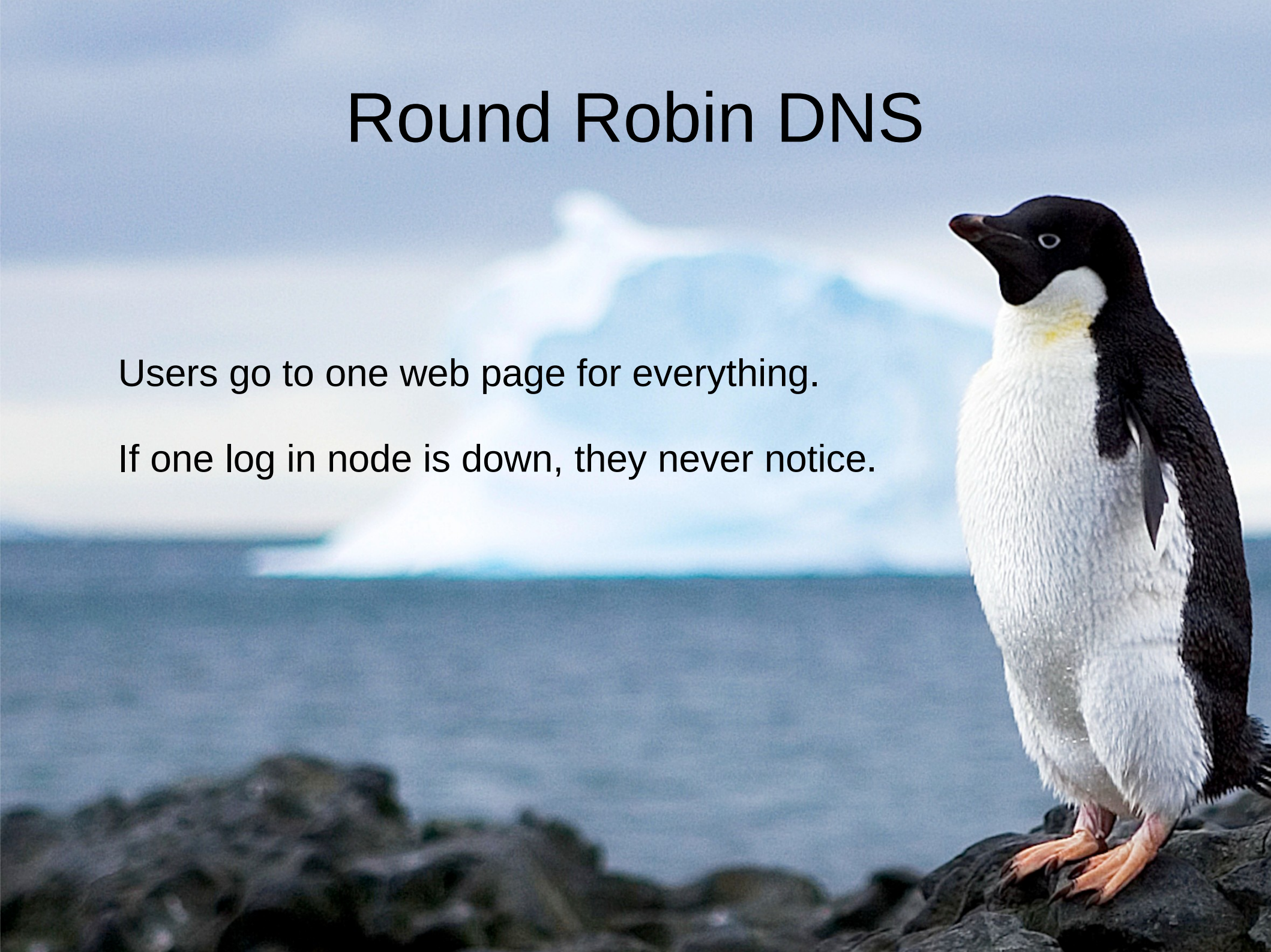

How do I fix this?

My architecture checklist

- Need a scheduler.
- Need to simplify the process.
- The user shouldn't know or care about what host they are on.
- Any distance between cluster complexity and the user is probably a good thing.



Round Robin DNS



Users go to one web page for everything.

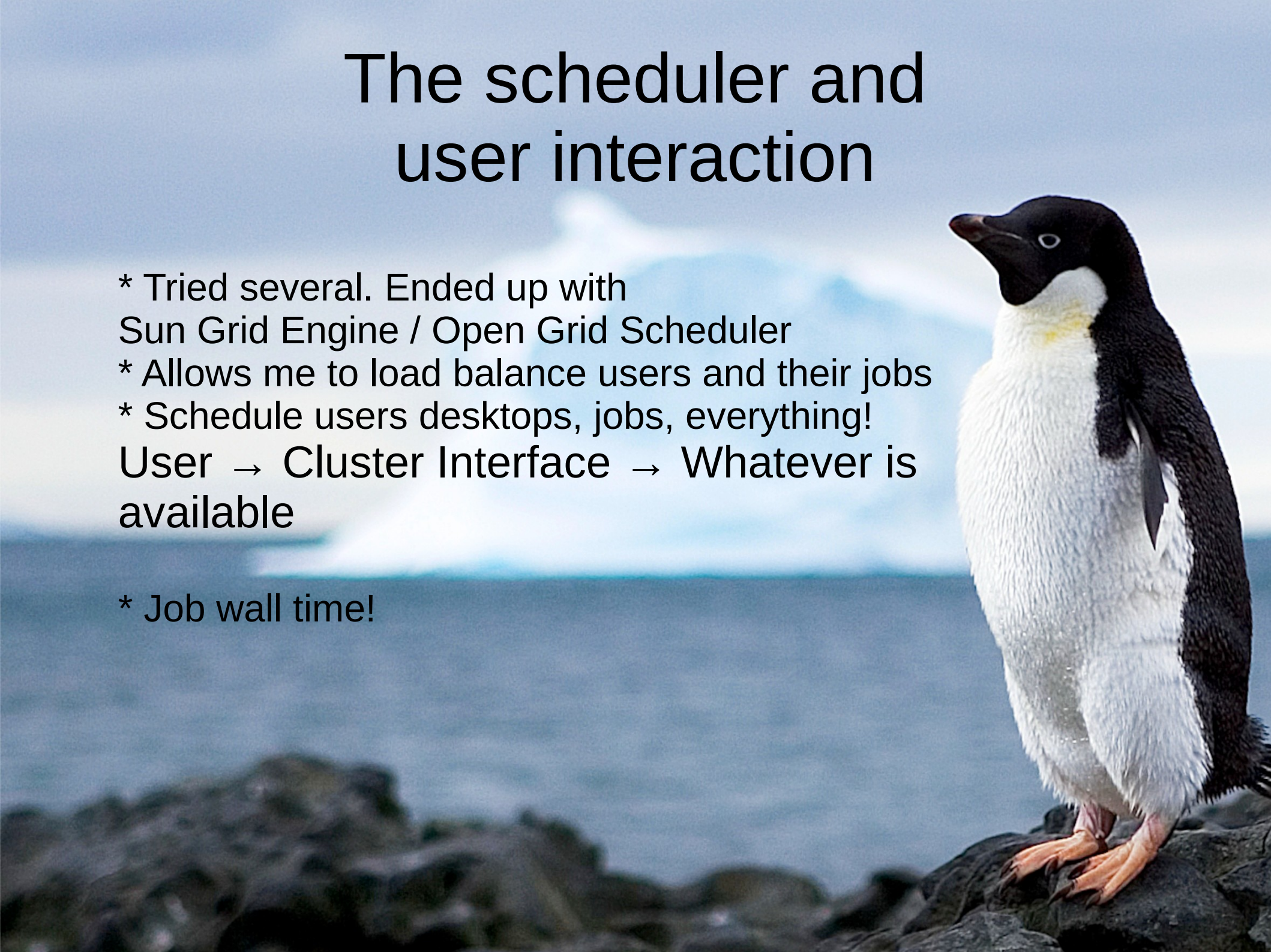
If one log in node is down, they never notice.

The scheduler and user interaction

* Tried several. Ended up with
Sun Grid Engine / Open Grid Scheduler
* Allows me to load balance users and their jobs
* Schedule users desktops, jobs, everything!

User → Cluster Interface → Whatever is available

* Job wall time!



I have a cluster and scheduler, now what?

Cluster <----------Big Gap----------> Users

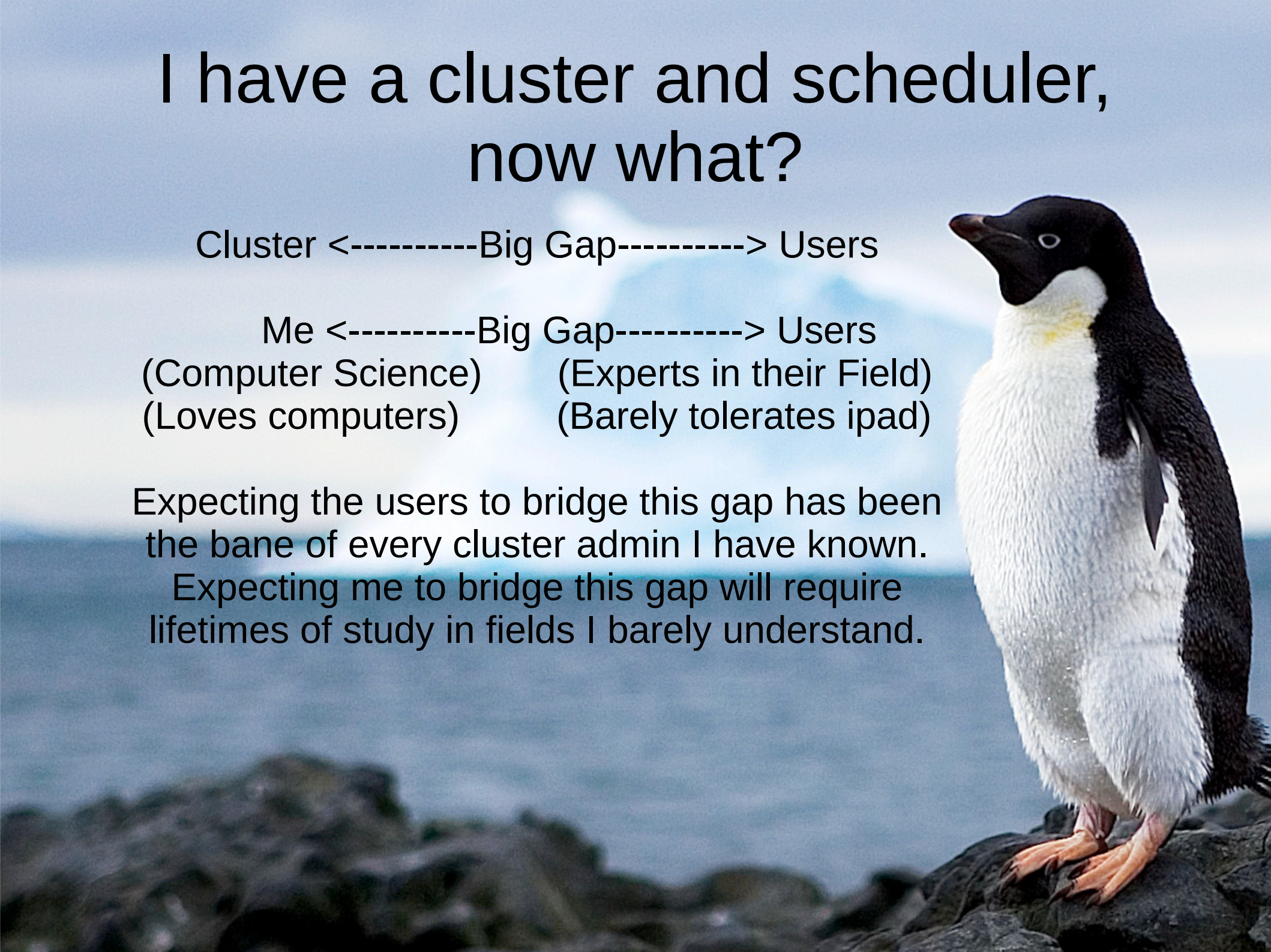
Me <----------Big Gap----------> Users

(Computer Science) (Experts in their Field)

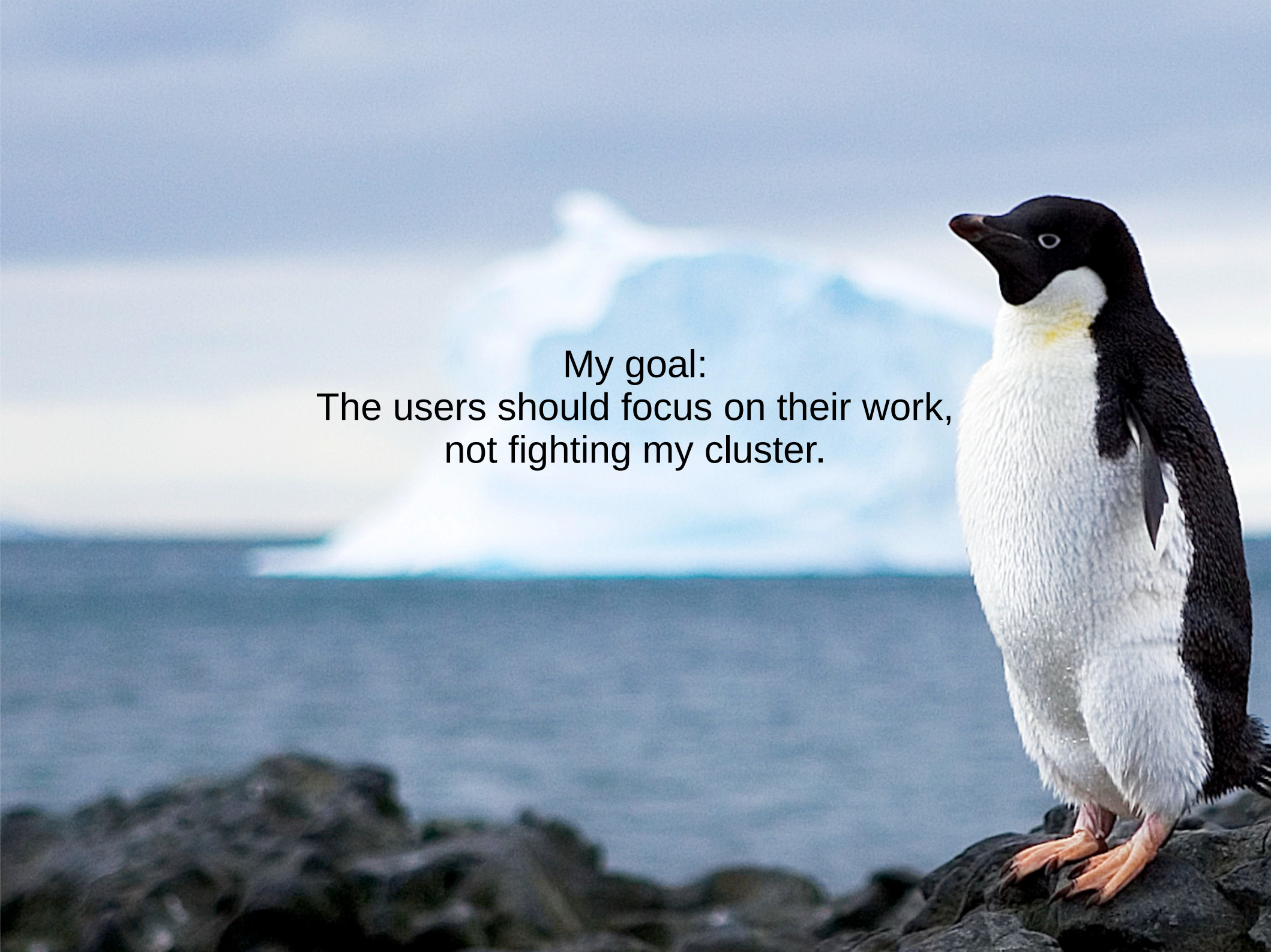
(Loves computers) (Barely tolerates ipad)

Expecting the users to bridge this gap has been the bane of every cluster admin I have known.

Expecting me to bridge this gap will require lifetimes of study in fields I barely understand.




What do I know?
The cluster.


My goal:
The users should focus on their work,
not fighting my cluster.

The journey begins

- First product: EnginFrame from Nice Software.
  - Tons of options | Lots of complexity
  - Very talented Engineering Support
  - Way more complex then we needed; things I didn't care about caused me grief
  - Gave me an excellent idea of where I wanted to go.

A challenger appears...

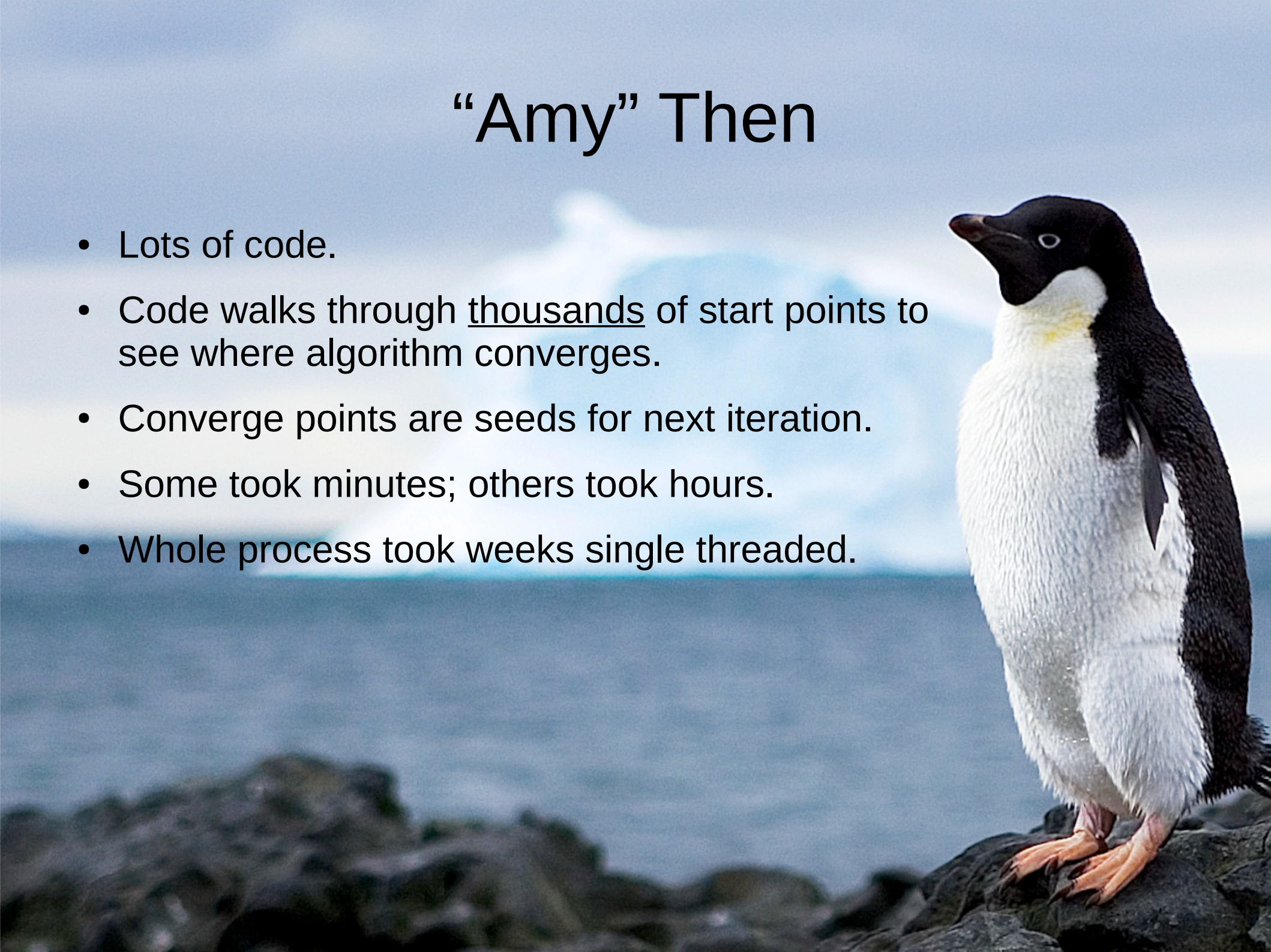
- eQUEUE from Advanced Clustering
  - Wraps together several products we were previously bundling individually.
  - Simple to understand
  - Simple to manage (Key=value style programming)
  - Simple way to interact with the cluster
  - BUILT IN HELP!!!!!

How does this help me reach my goal?

- We personally walk through the cluster with every new user. An introduction. We expect users to ask follow up questions, but a personal introduction saves time and effort for both admins and users.
- We can now craft interfaces to simplify their interaction with the cluster. We do this by by seeing their workflow and adapting the interface to match and automate their needs.

“Amy” Then

- Lots of code.
- Code walks through thousands of start points to see where algorithm converges.
- Converge points are seeds for next iteration.
- Some took minutes; others took hours.
- Whole process took weeks single threaded.



“Amy” Now

- Simple interface for her to use
- Simple interface for us to manage
- Uses full cluster resources through scheduler
- And most importantly, she can now run bigger data sets over FAR more iterations because the “easy” parallelization given through the “easy” interface afford her the time to do so.
- What would have been /weeks/ is now a day.

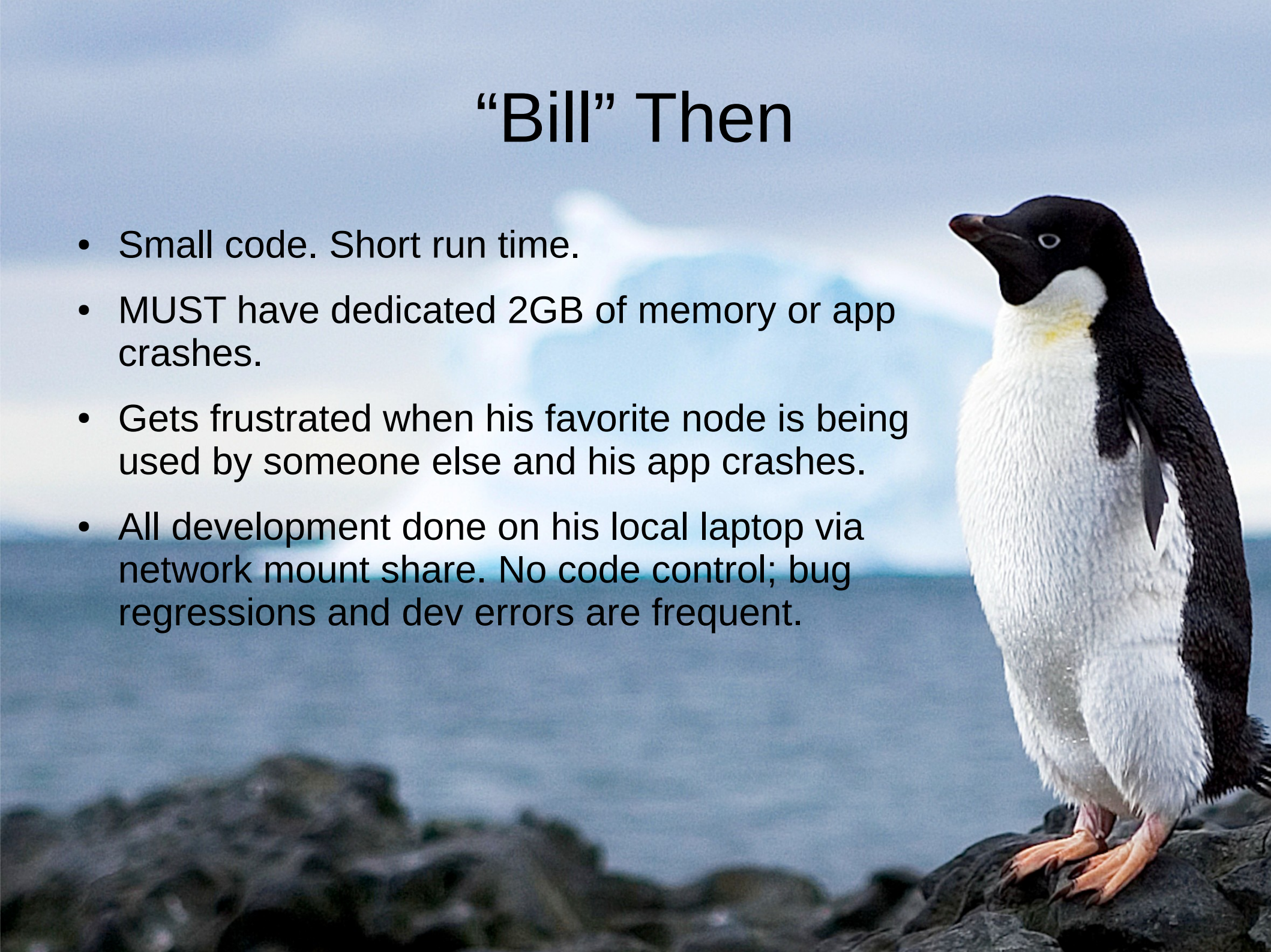
Does Amy fit the goal?

- She spends more time making sense of the data and writing papers on her findings then she does waiting for her job to finish on the cluster by a WIDE margin.

I think so.

“Bill” Then

- Small code. Short run time.
- MUST have dedicated 2GB of memory or app crashes.
- Gets frustrated when his favorite node is being used by someone else and his app crashes.
- All development done on his local laptop via network mount share. No code control; bug regressions and dev errors are frequent.



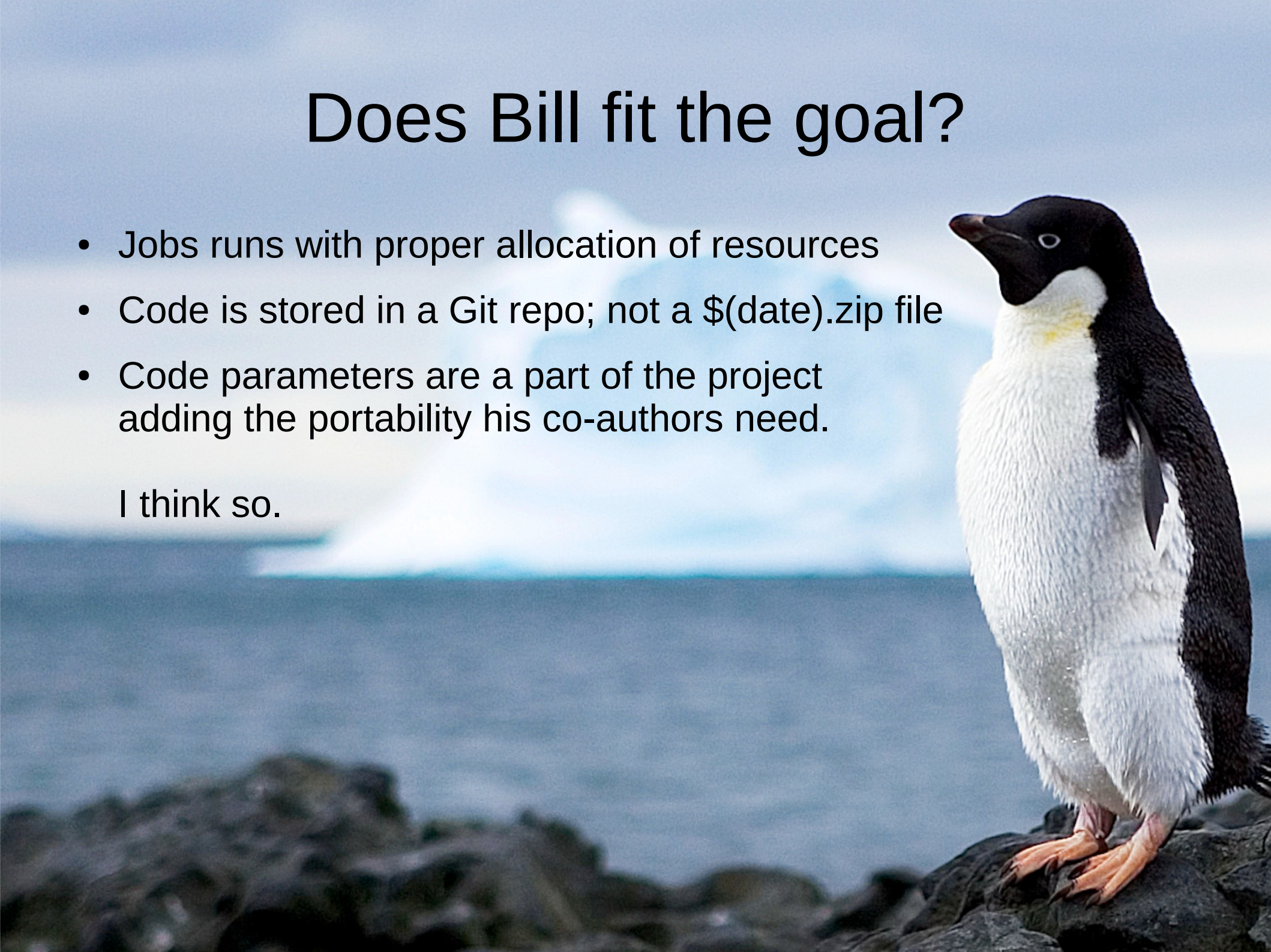
“Bill” Now

- Simple web interface.
- Pulls latest code from Git repo. Reads the configuration and compiles appropriately.
- Doesn't care where his job runs as it is handled via the scheduler.

Does Bill fit the goal?

- Jobs runs with proper allocation of resources
- Code is stored in a Git repo; not a $(date).zip file
- Code parameters are a part of the project adding the portability his co-authors need.

I think so.




Whose next?


Questions?