Scaling Gerrit
The Good, The Bad and The Ugly
Agenda

› Background
› Briefly where are we now with Gerrit
› Stretching Gerrit – Case study
› Vertical & Horizontal Scaling
› JGit
› A few open questions
Background

Ericsson is a federated company, with no single way of working.

- Project full ownership is delegated:
  - Easy to scale.
  - Empowers and encourages people to build expertise.
Where are we now

Gerrit scales well in general:
- One master instance
- 14K monthly unique users
- 12K+ repos hosted
- 1M changes
- 3-6K new changes/day
- About 20K pushes/day
- Up to 140K queries/day
- Replication - 7 geographically distributed slave clusters
Case Study – Team

Busy repositories:

- Close to 2K geographically distributed team
- Using submodules
- Most changes are done in super repo
- Use code review exclusively
Case Study– Repository

› Busy repo:
  ▪ Private/disposable branches – stable 8K-10K refs
  ▪ 300K refs, 100K refs last 6 months
  ▪ 4.1 Million objects
  ▪ 18MB packed refs
  ▪ 67M bitmap (JGit)
Case Study – Stretching

Stretching Gerrit:
- Extensive automation
- Longer lasting transactions
- JGit gc runs multiple times a day
- Risks of bottlenecks

How multi-master would handle such a use case?
Heap memory management

› Slaves horizontally scaled
› Master - Vertical scaling + hot standby
› Master java heap 58GB

› A cluster of masters to have Gerrit fully cloud-friendly
Repo corruption:

- Lost commit for cherry pick and rebase if necessary [1]
- Unable to push with missing sha-1 [2][3]
- Delete objects on garbage collection (solved)[4]
Wrapping up

› Busy repos and Gerrit compatibility on long term
› Vertical & Horizontal scaling
› JGit reliability
Open Questions

› How will notedb scale for same busy repos?
› How reliable is to store all metadata in git (JGit context)?
› How could multi-master handle the presented use case?
Reference
