WALT on Grid'5000 (with NBFS)

or how to pack all my activities into a single project ;)

Etienne Dublé (LIG / CNRS)

Workshop Axes, May 2021

Agenda

Motivation

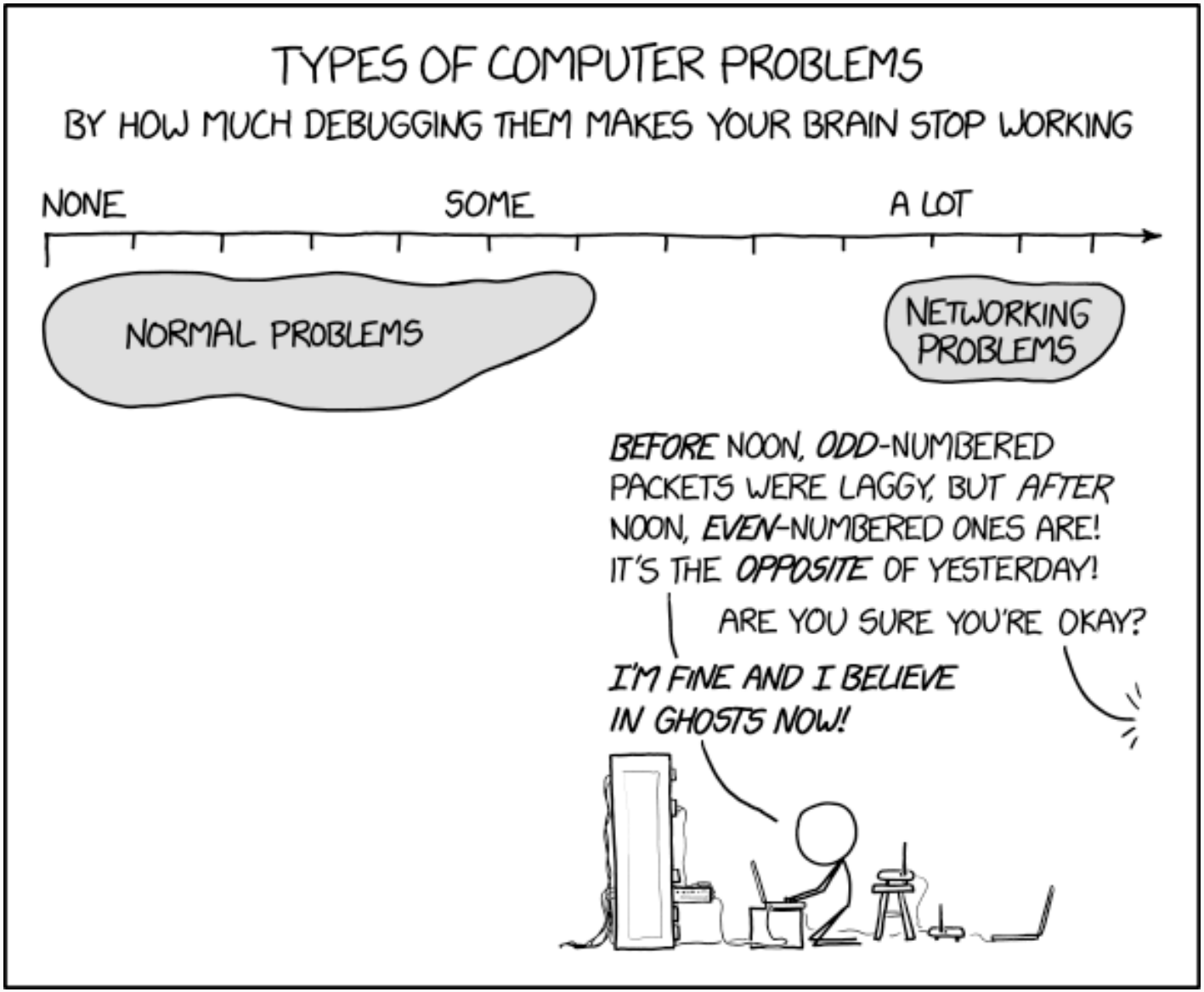
Sample use case

WalT & fast OS prototyping

OS deployment rework

Related news

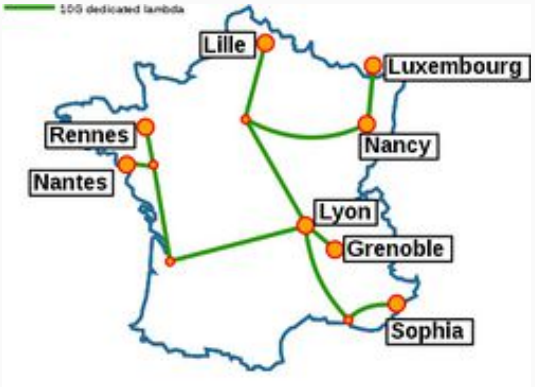
Foreword


TYPES OF COMPUTER PROBLEMS
BY HOW MUCH DEBUGGING THEM MAKES YOUR BRAIN STOP WORKING
NONE
SOME
A LOT
NORMAL PROBLEMS
NETWORKING PROBLEMS
BEFORE NOON, ODD-NUMBERED PACKETS WERE LAGGY, BUT AFTER NOON, EVEN-NUMBERED ONES ARE! IT'S THE OPPOSITE OF YESTERDAY!
ARE YOU SURE YOU'RE OKAY?
I'M FINE AND I BELIEVE IN GHOSTS NOW!

A word about Grid'5000

G5K: What is it?

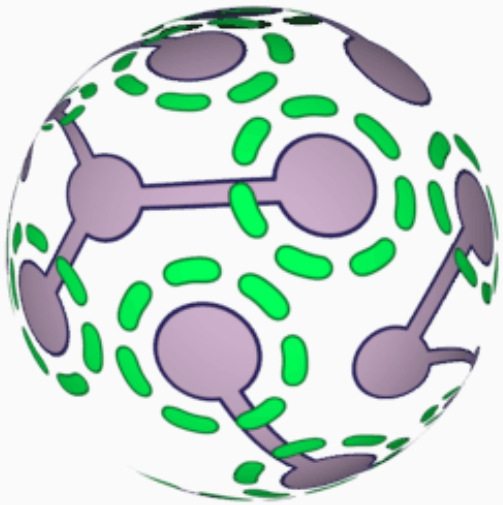
- Infrastructure as a Service
- 8 sites in France
- Accessible to academic community



A word about WALT

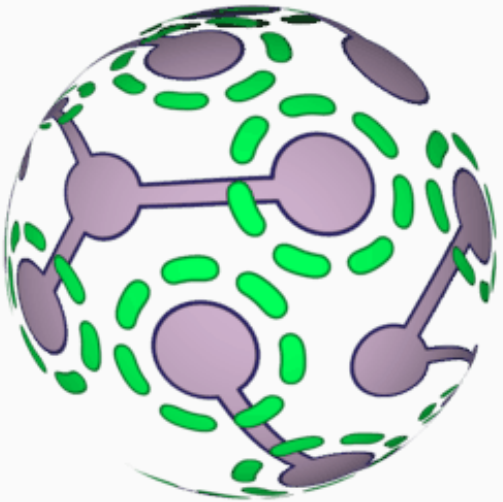
WALT: What is it?

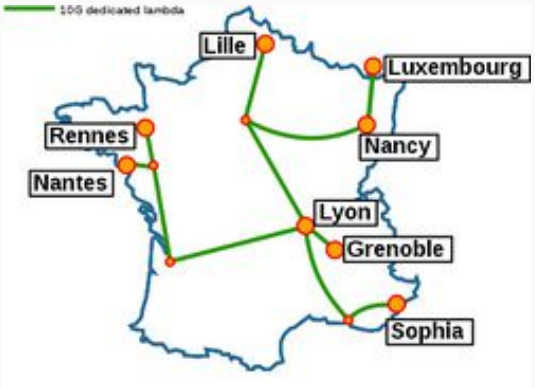
- Software to build your own versatile platform
- AFAIK deployed in 4 labs (France, Turkey) and at Schneider Electric
- Open source project



Why WALT on G5K?

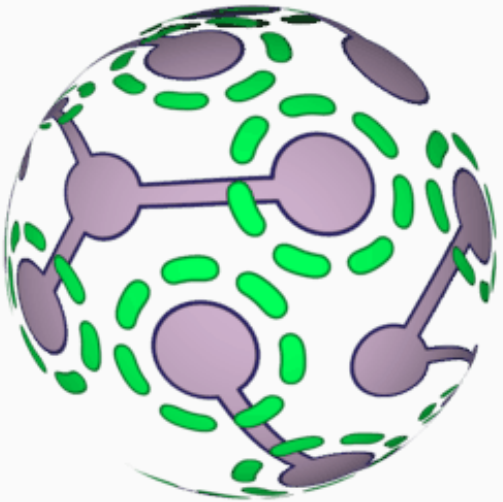
- Why WALT on G5K?





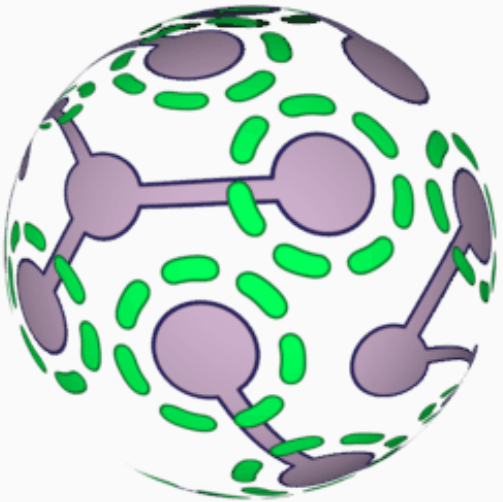
Why WALT on G5K?

- Why WALT on G5K?
  1. "I want to run my WALT experiment unmodified on a larger testbed"



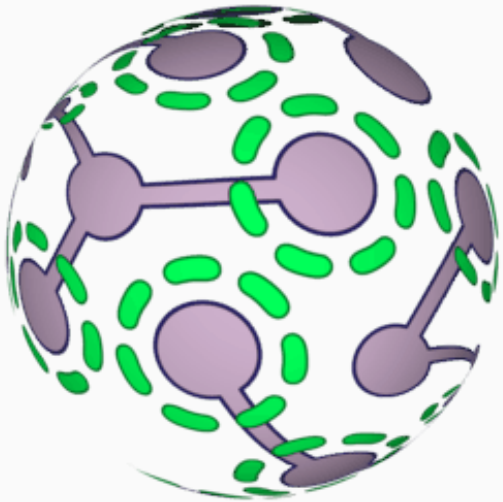
Why WALT on G5K?

- Why WALT on G5K?
  1. "I want to run my WALT experiment unmodified on a larger testbed"
  2. WALT could bring some useful features on top of G5K
     - **fast OS prototyping**
     - **seamless multi-site experiments**



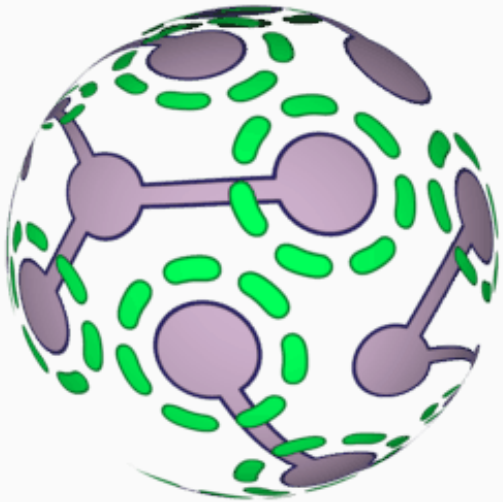
Why WALT on G5K?

- Why WALT on G5K?
  1. "I want to run my WALT experiment unmodified on a larger testbed"
  2. WALT could bring some useful features on top of G5K
     - **fast OS prototyping**
     - **seamless multi-site experiments**
  3. It could be an entrypoint for new users who want to try WALT



Why WALT on G5K?

- Why WALT on G5K?
  1. "I want to run my WALT experiment unmodified on a larger testbed"
  2. WALT could bring some useful features on top of G5K
     - **fast OS prototyping**
     - **seamless multi-site experiments**
  3. It could be an entrypoint for new users who want to try WALT
- Limits:
  - Features are limited to what both platforms can do (simplified G5K resources selection, wired network only, no physical access to devices, etc.)



Agenda

Motivation

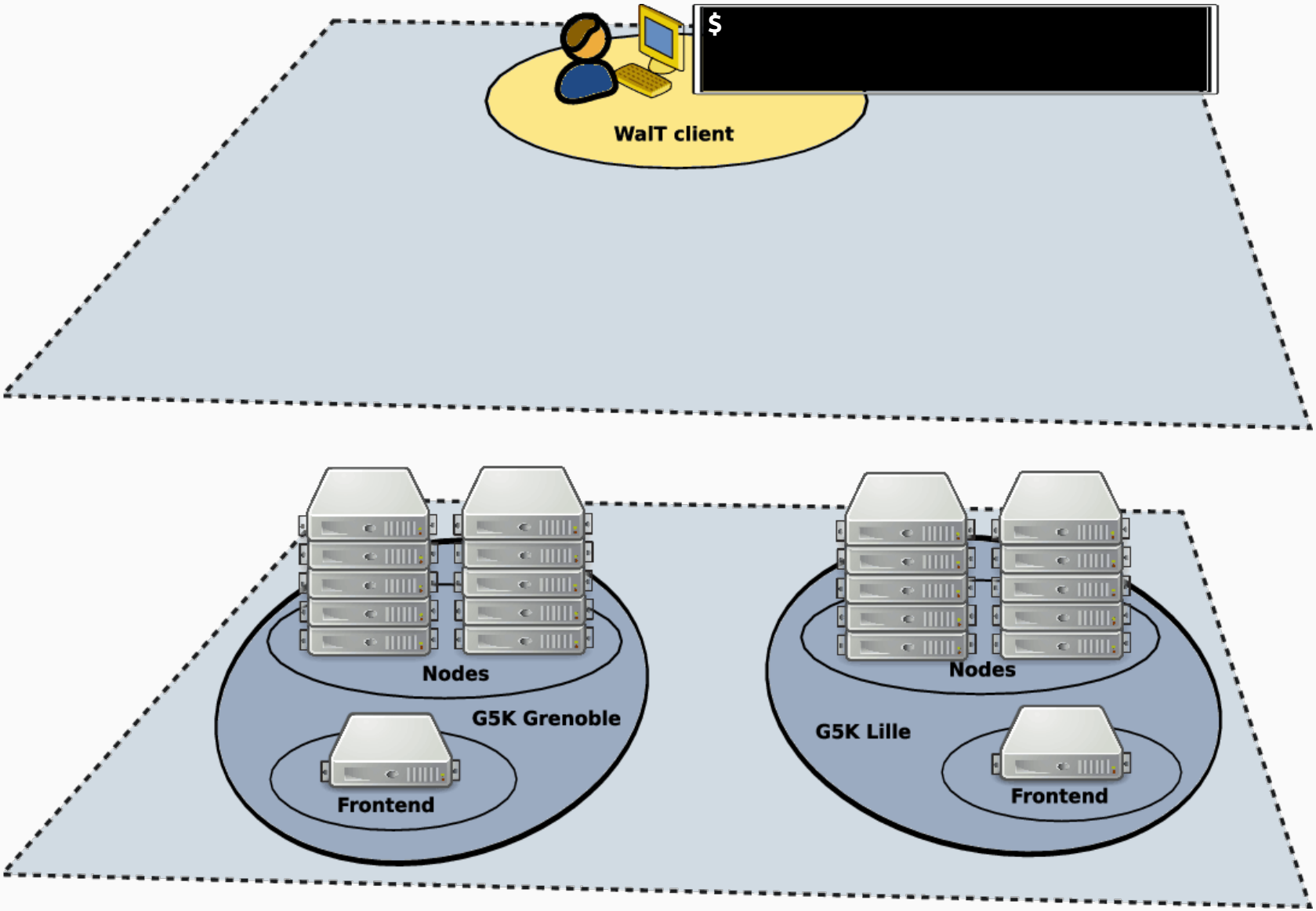
Sample use case

WalT & fast OS prototyping

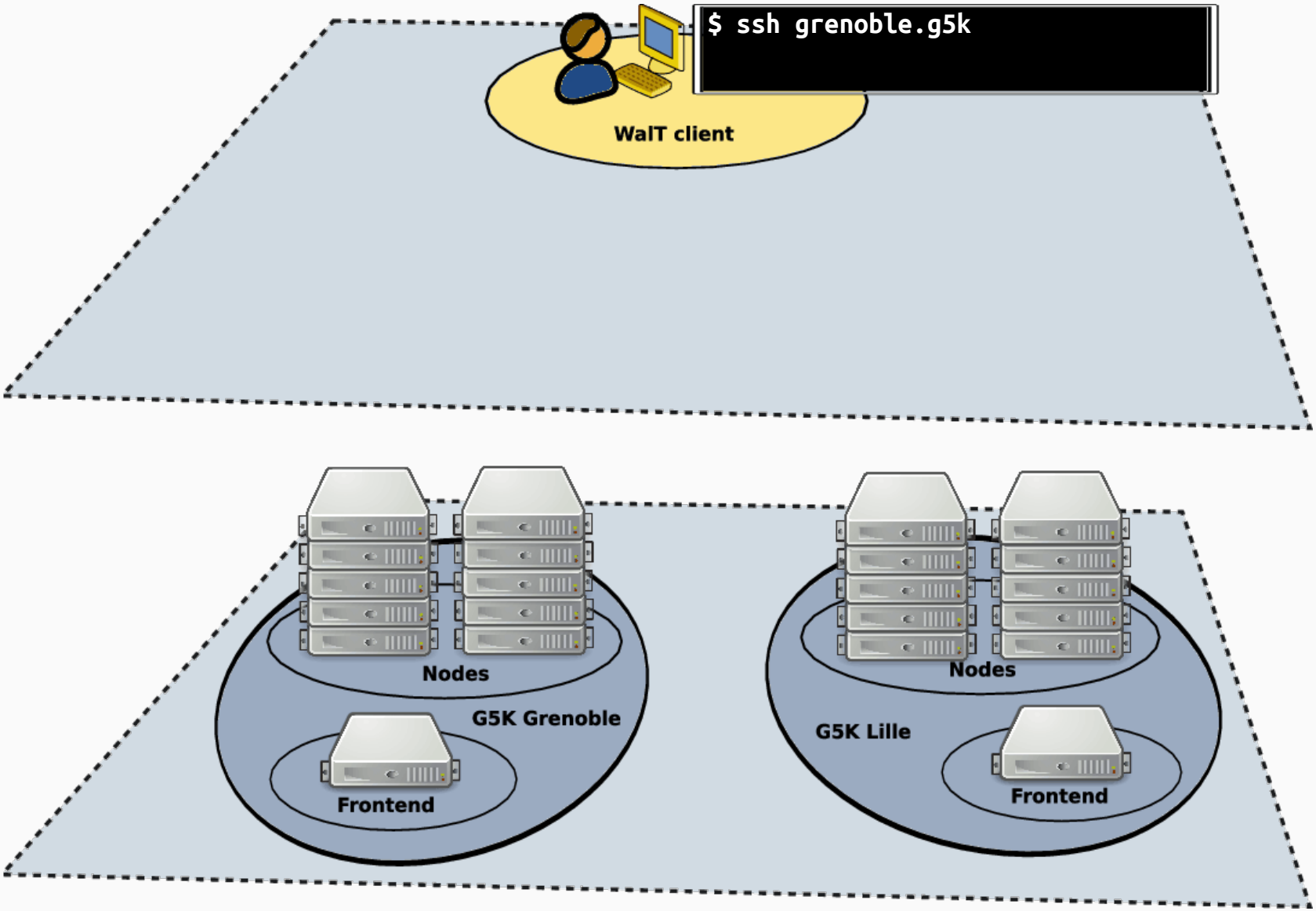
OS deployment rework

Related news

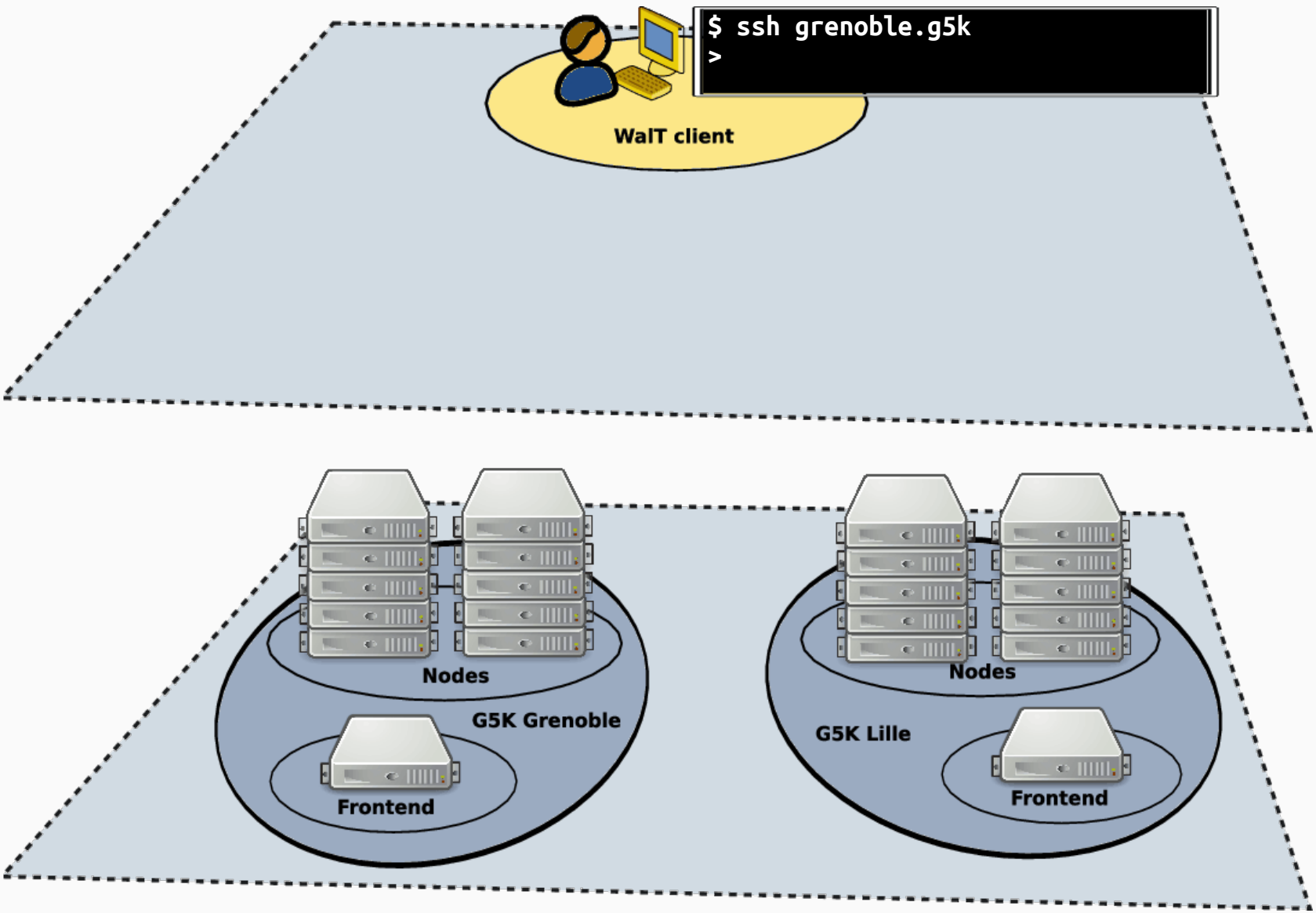
Sample use case



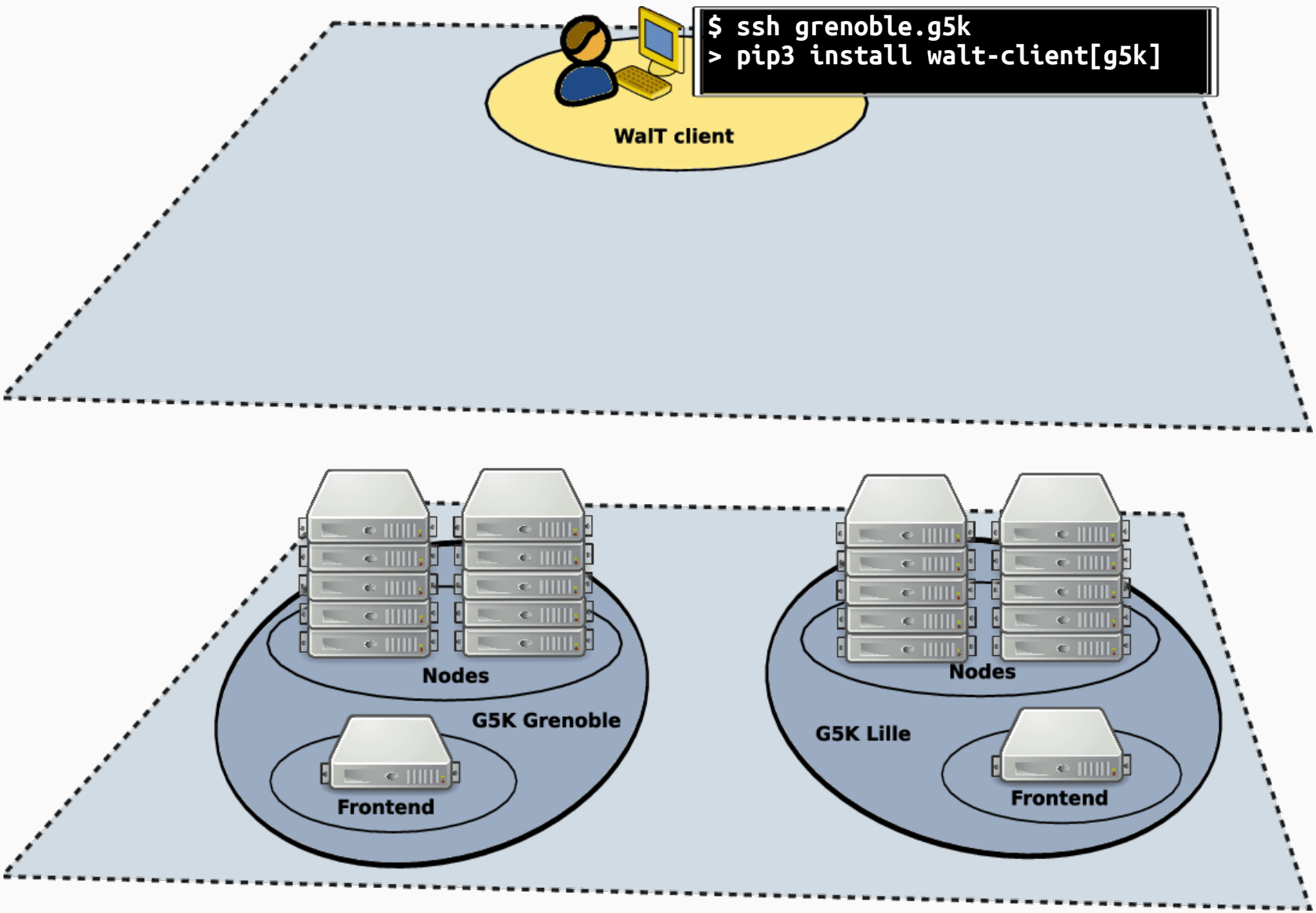
Sample use case



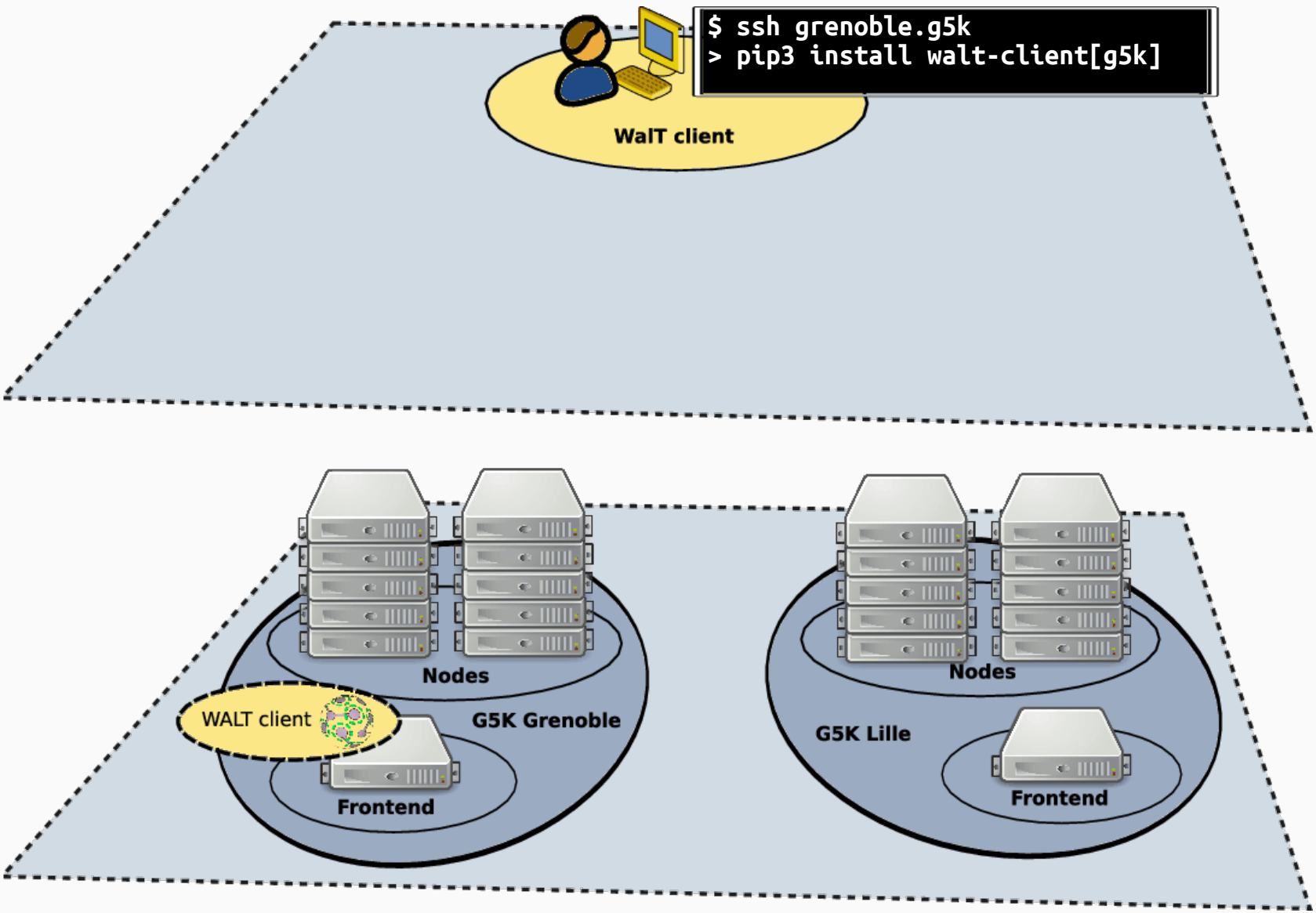
Sample use case



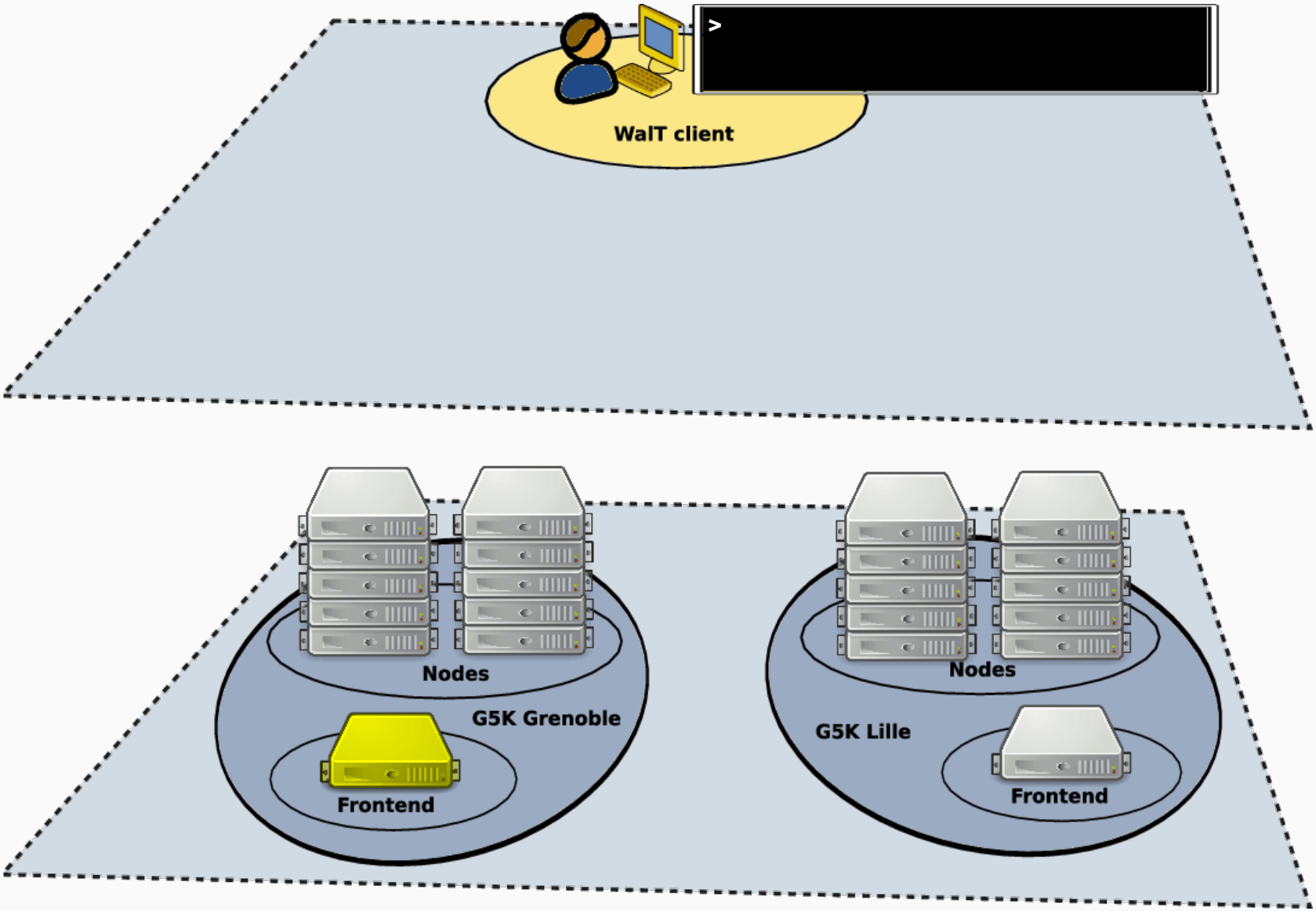
Sample use case



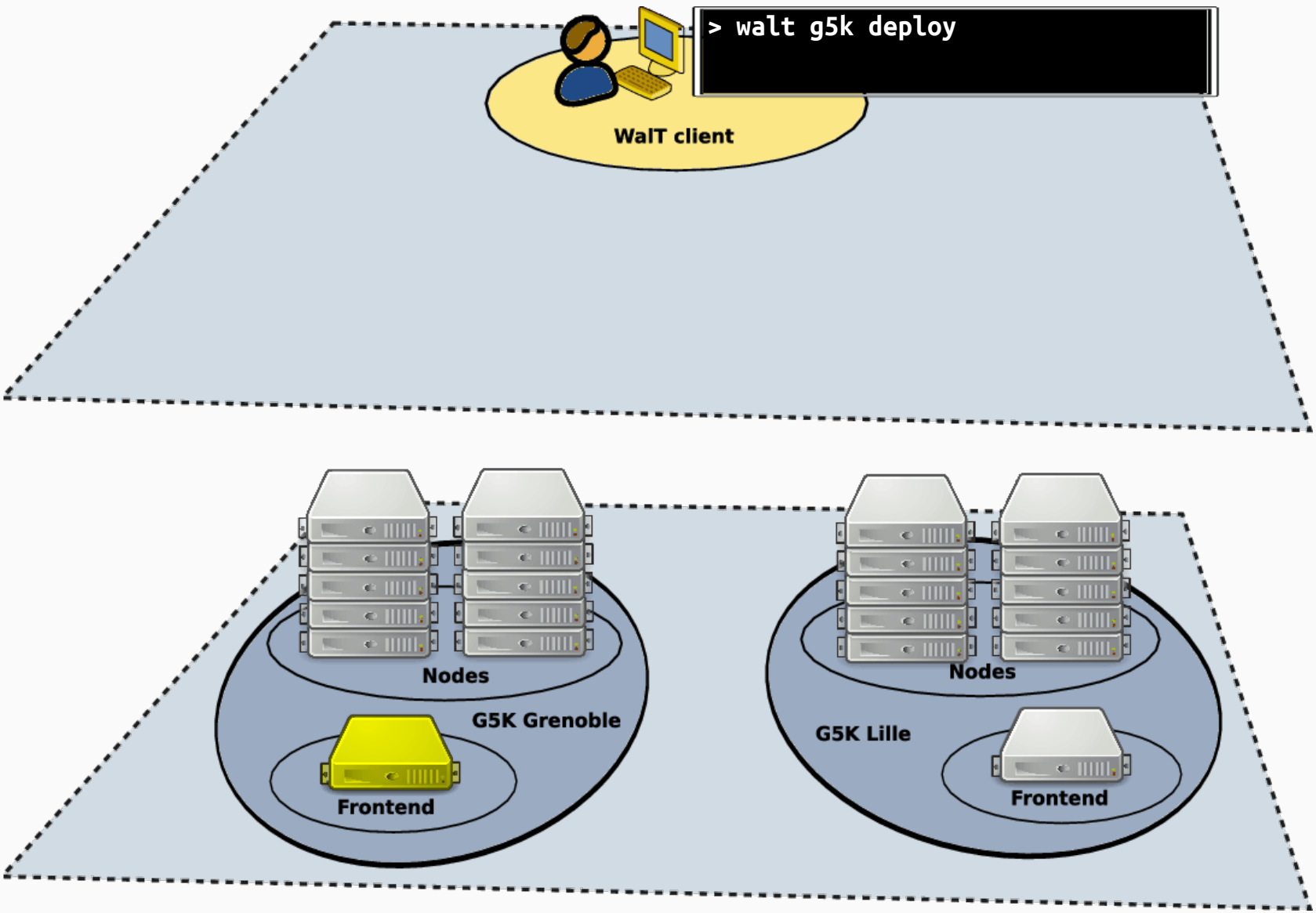
Sample use case



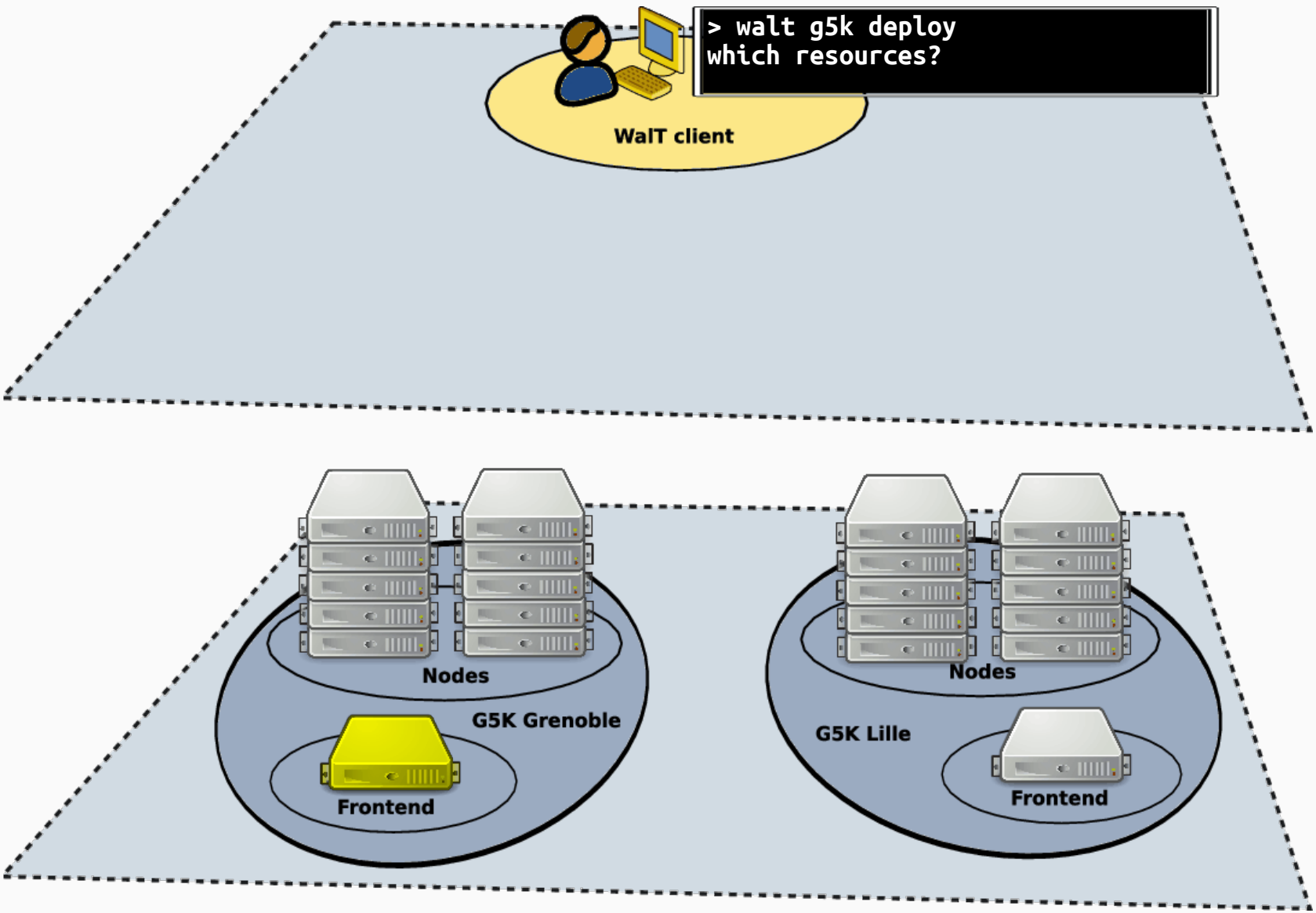
Sample use case



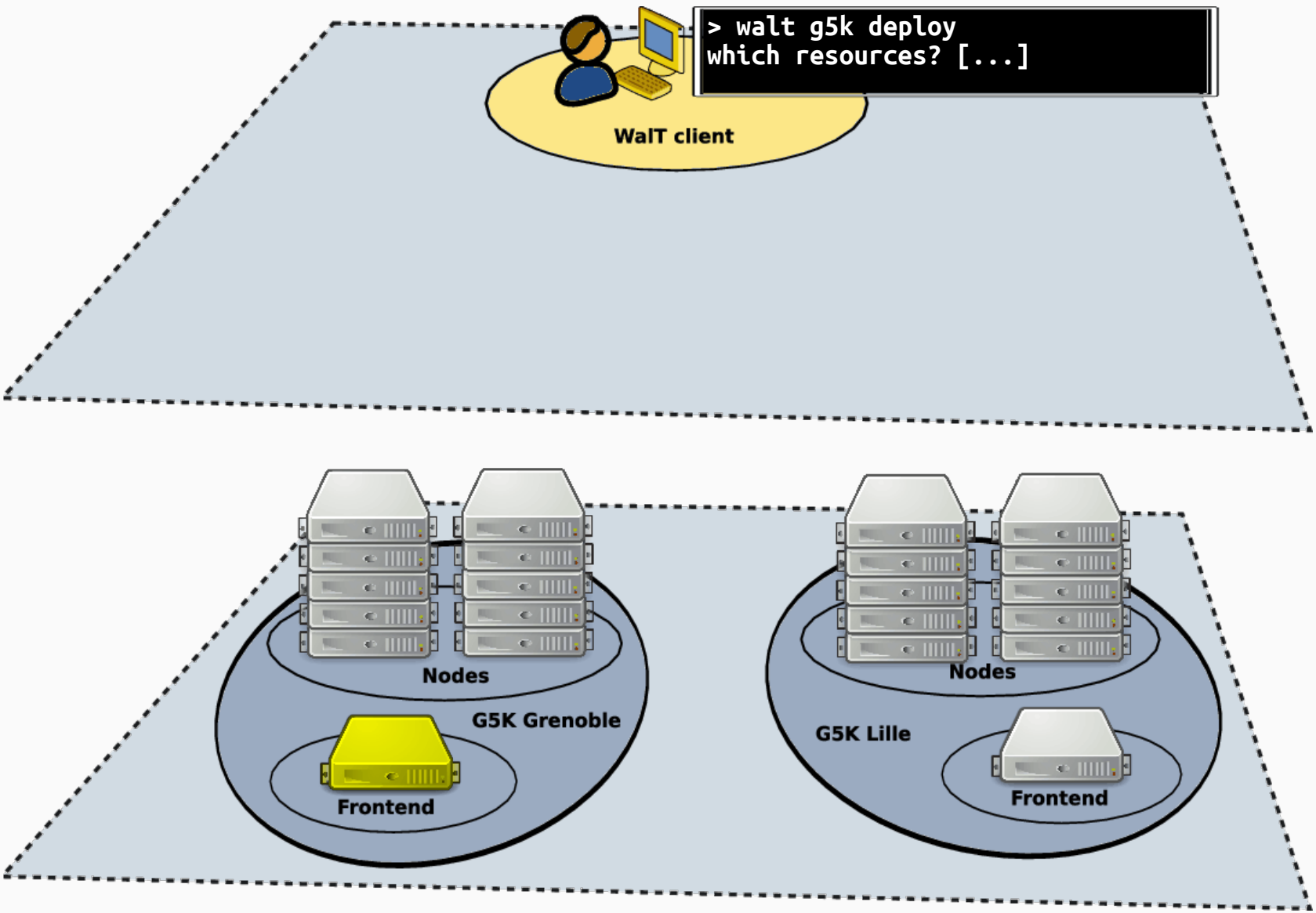
Sample use case



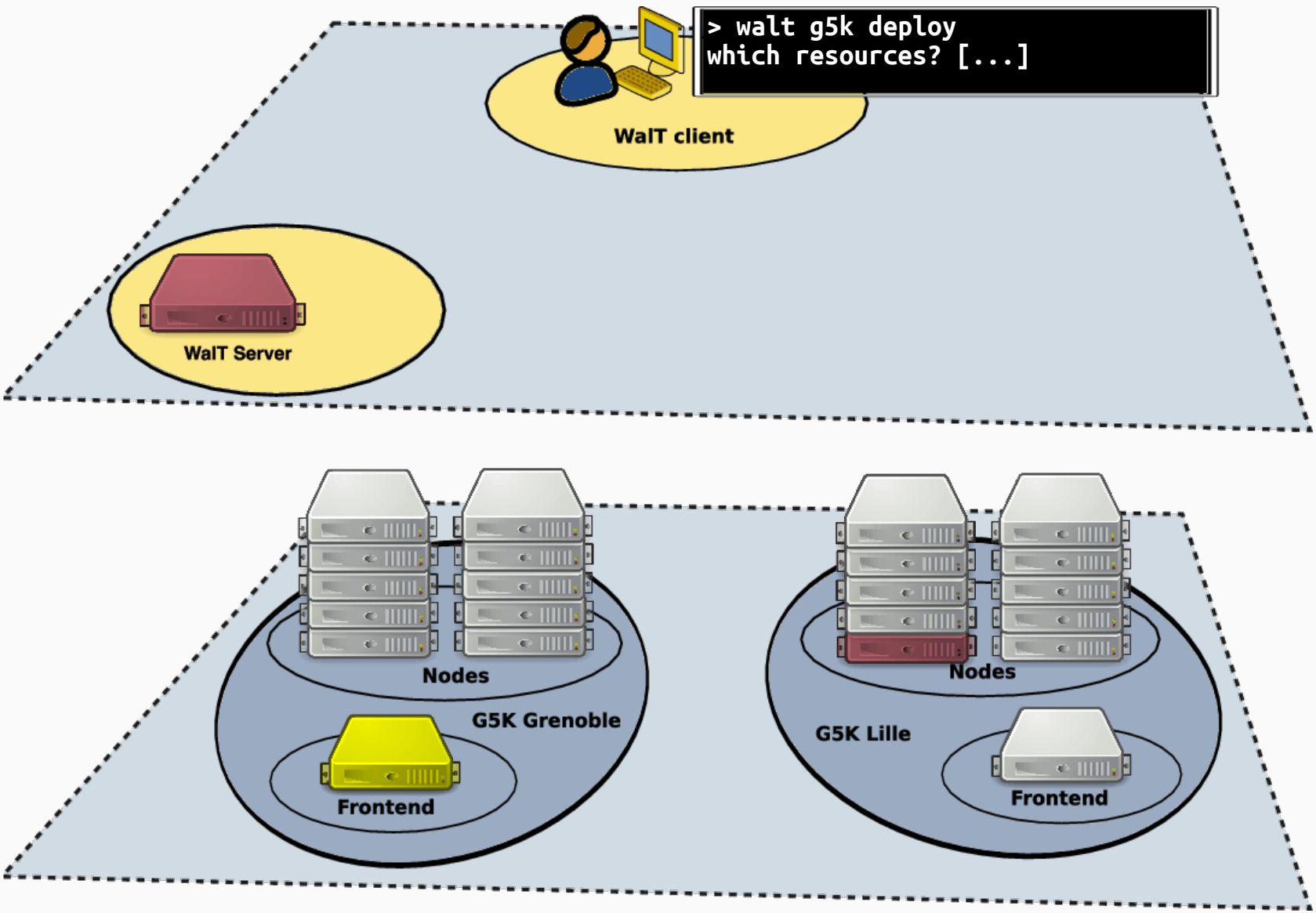
Sample use case



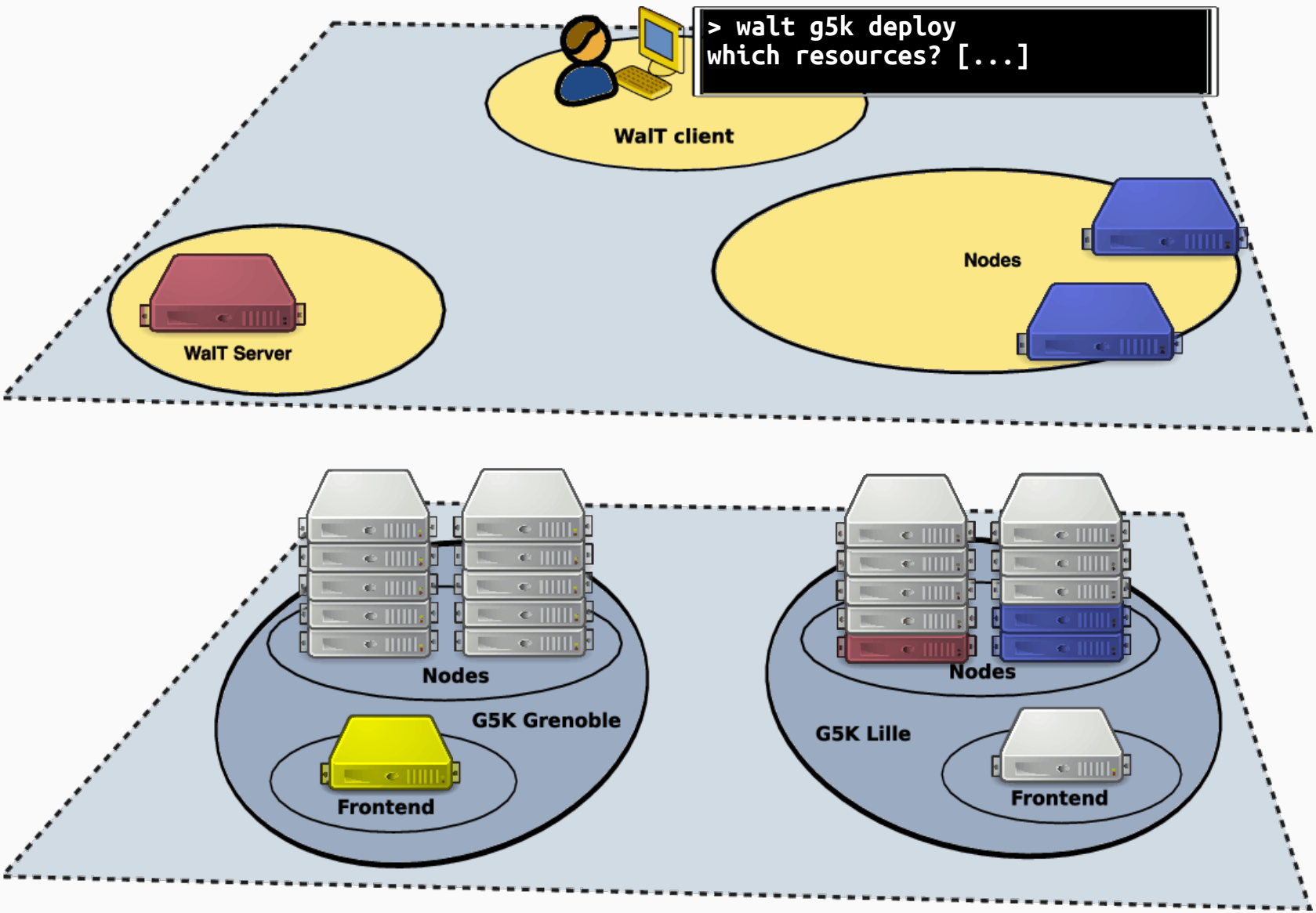
Sample use case



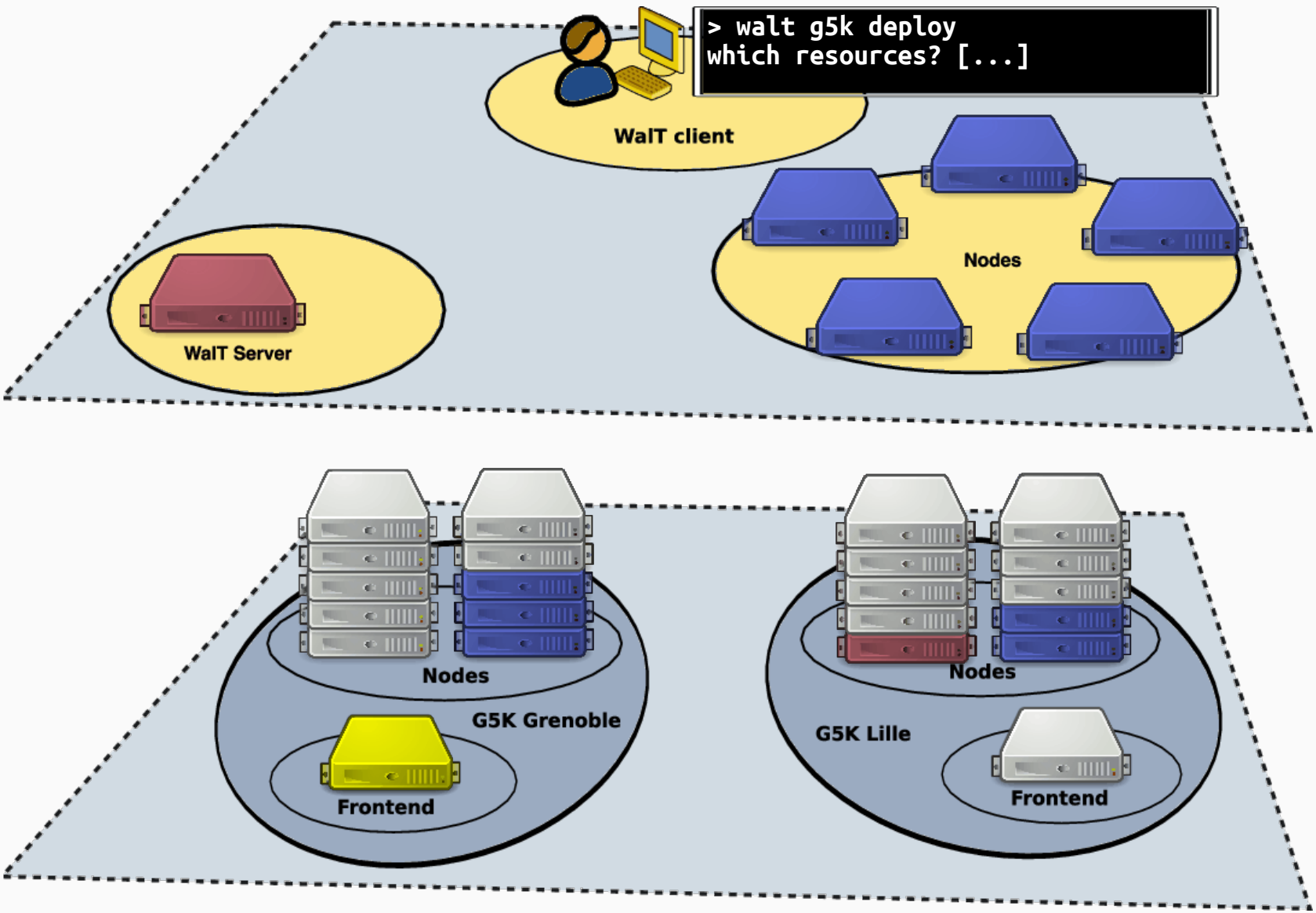
Sample use case



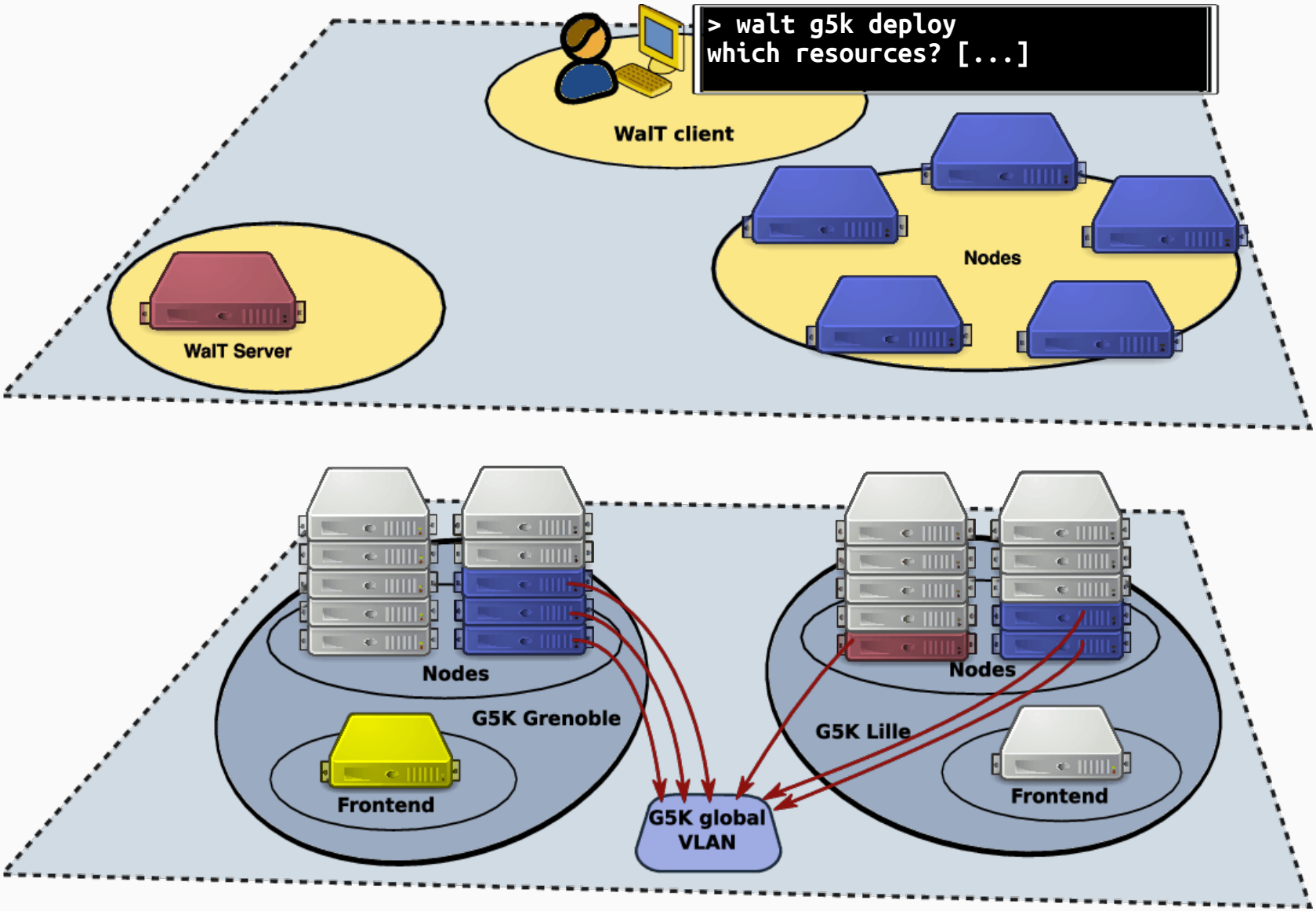
Sample use case



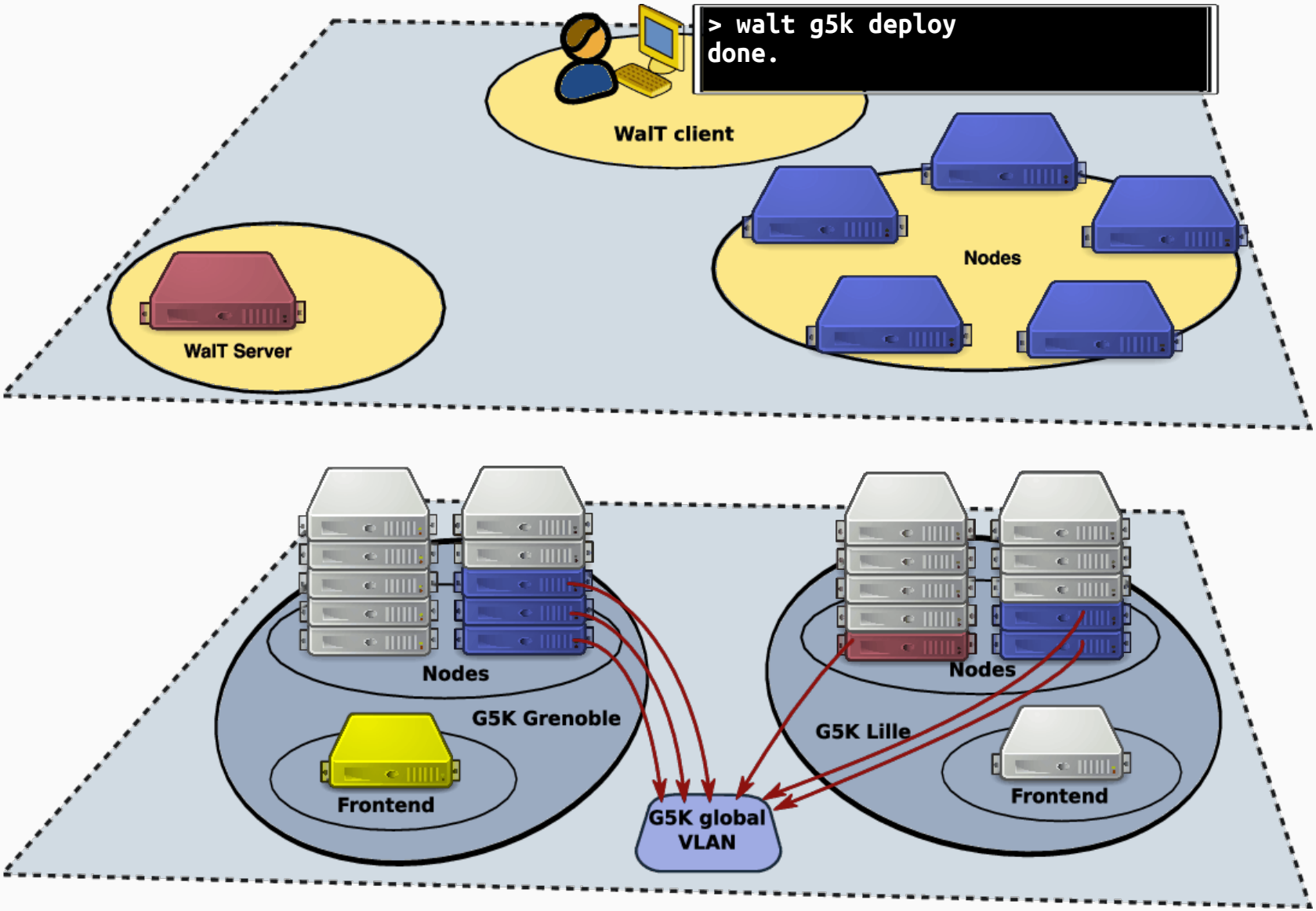
Sample use case



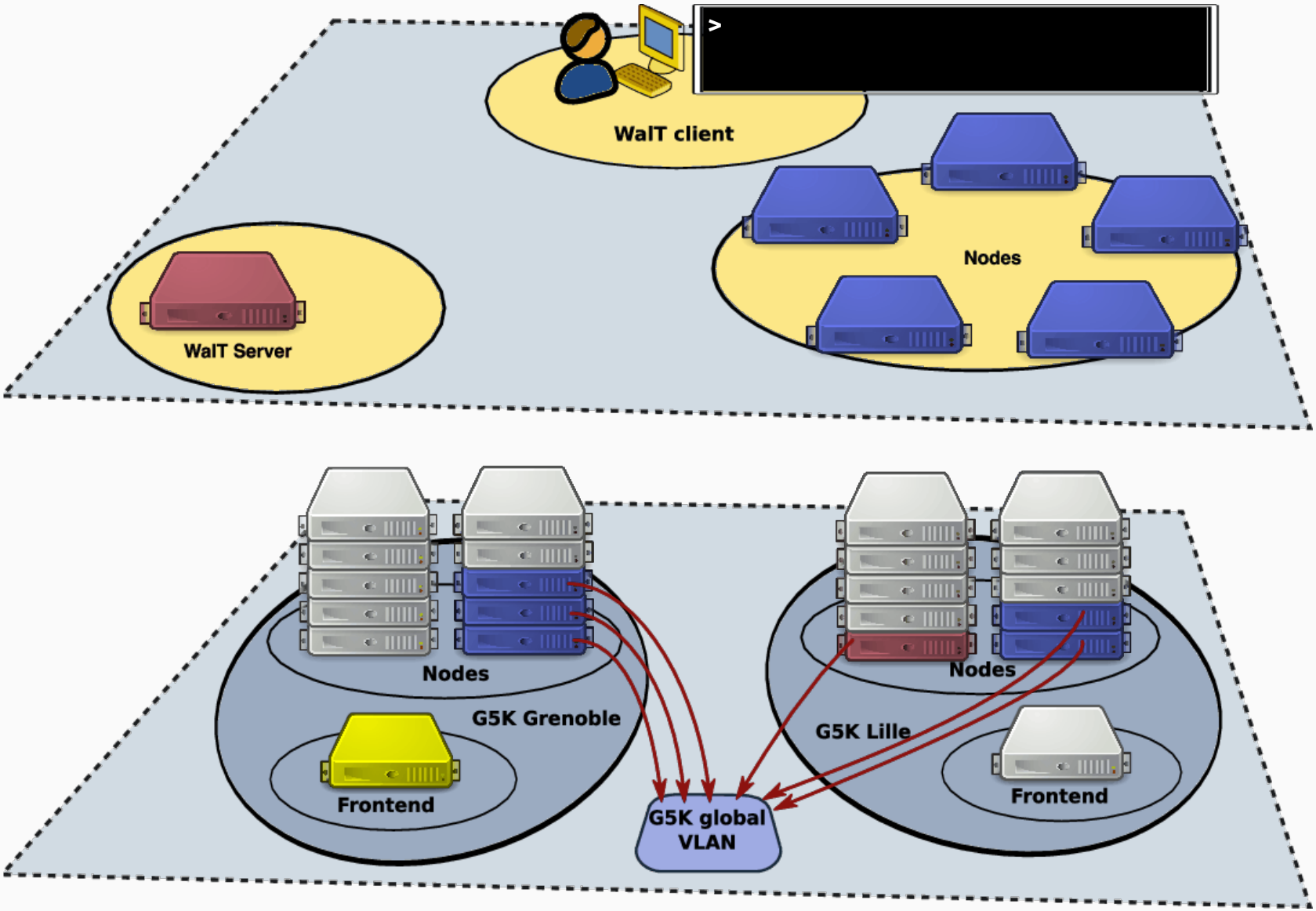
Sample use case



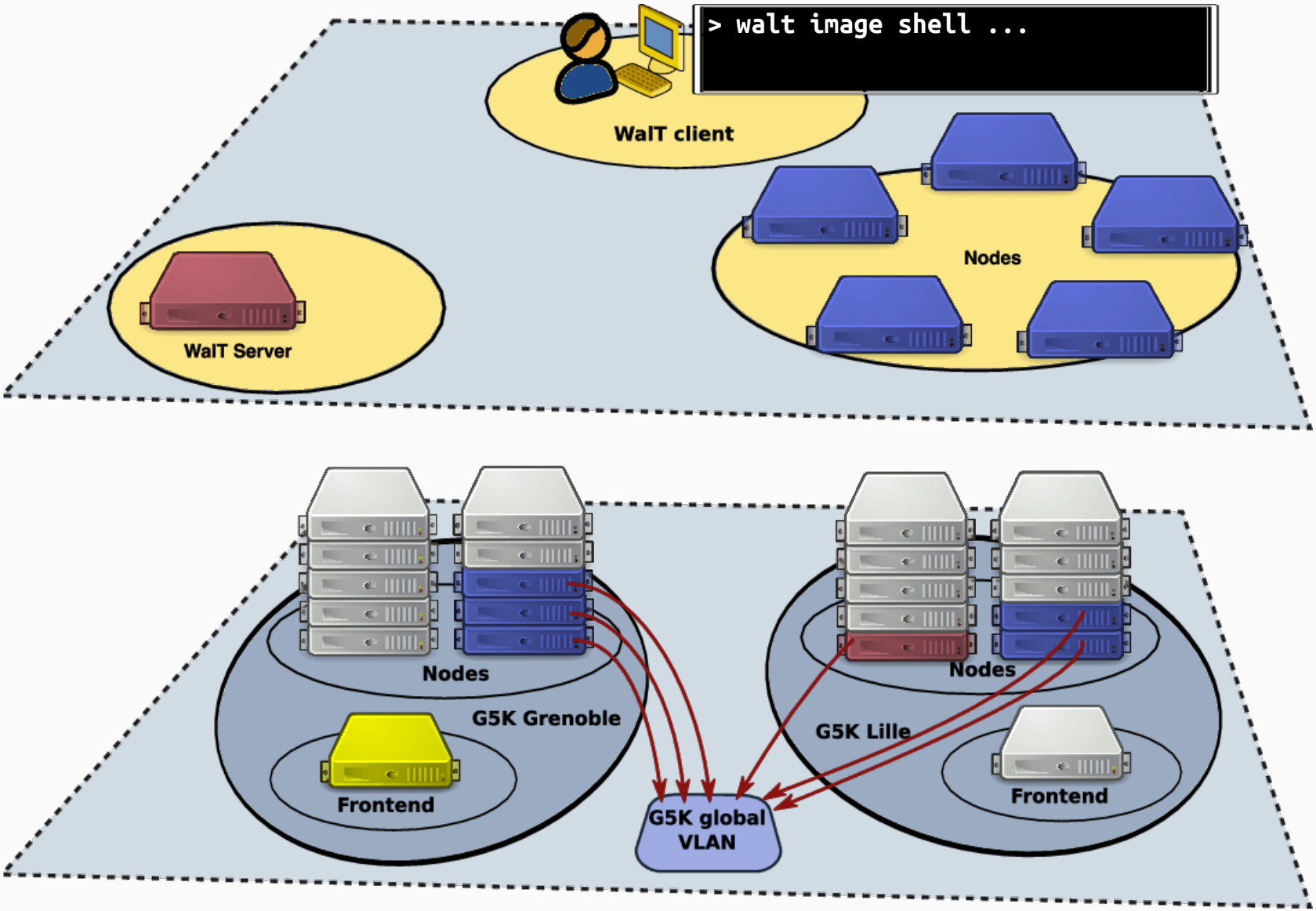
Sample use case



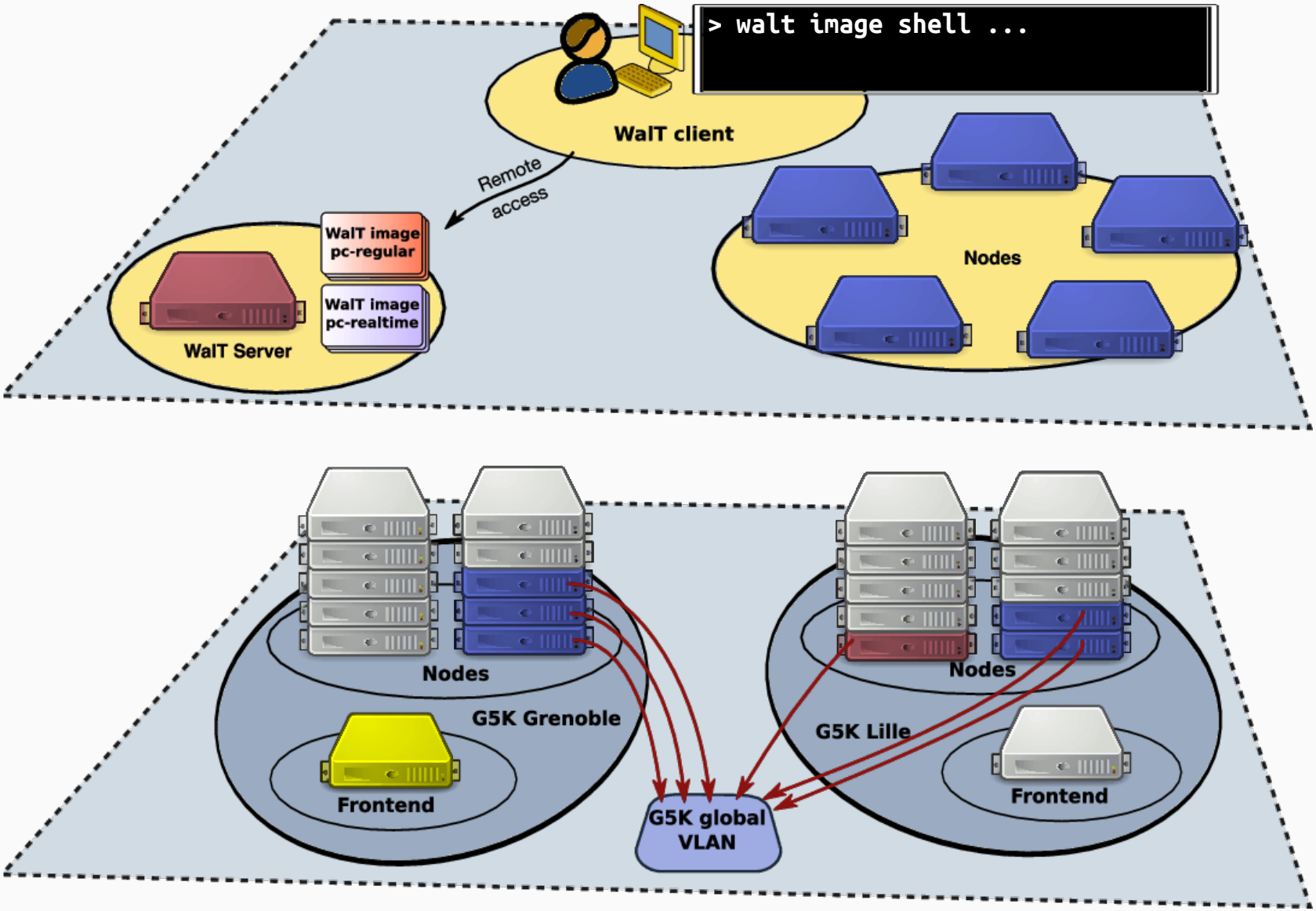
Sample use case



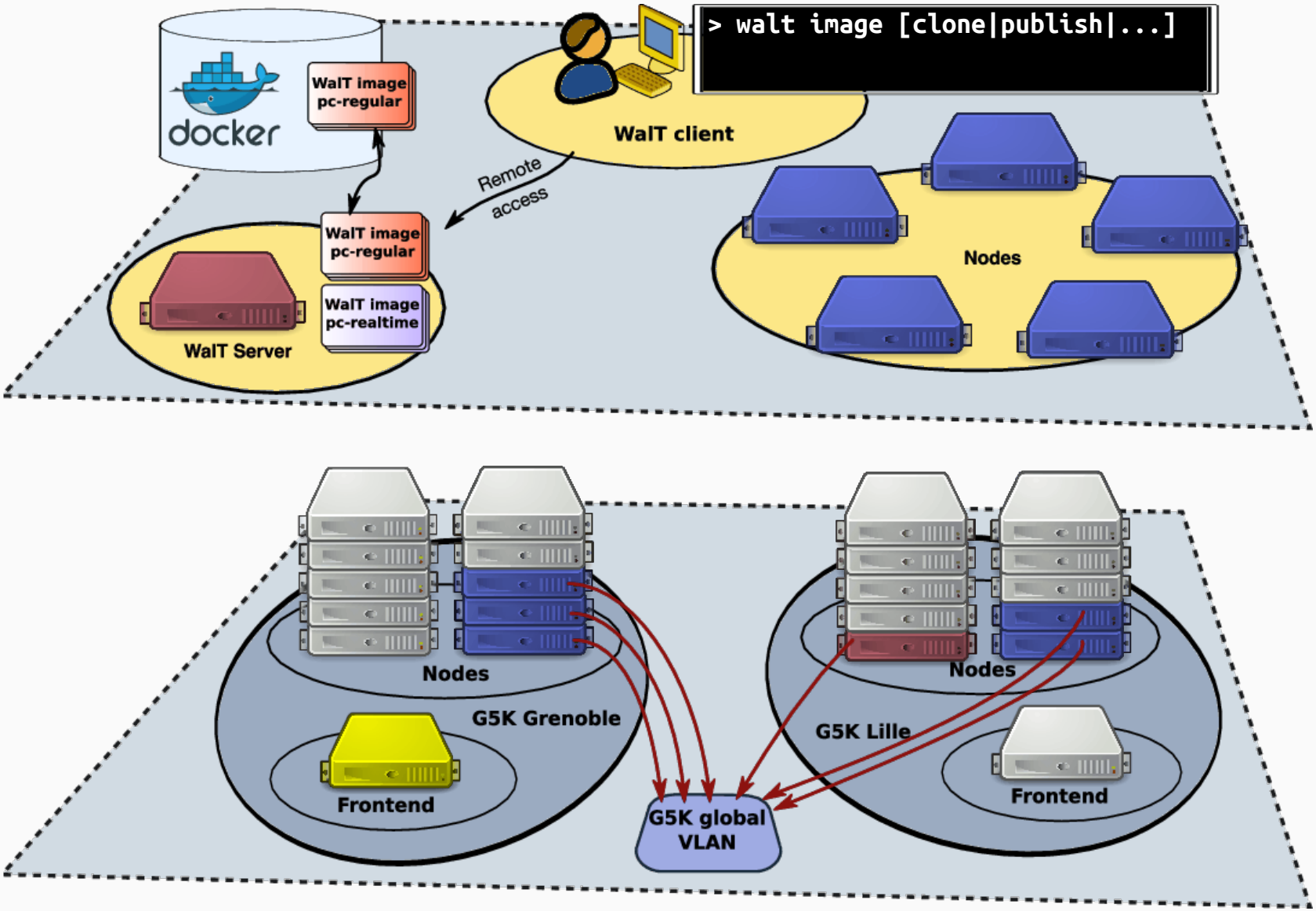
Sample use case



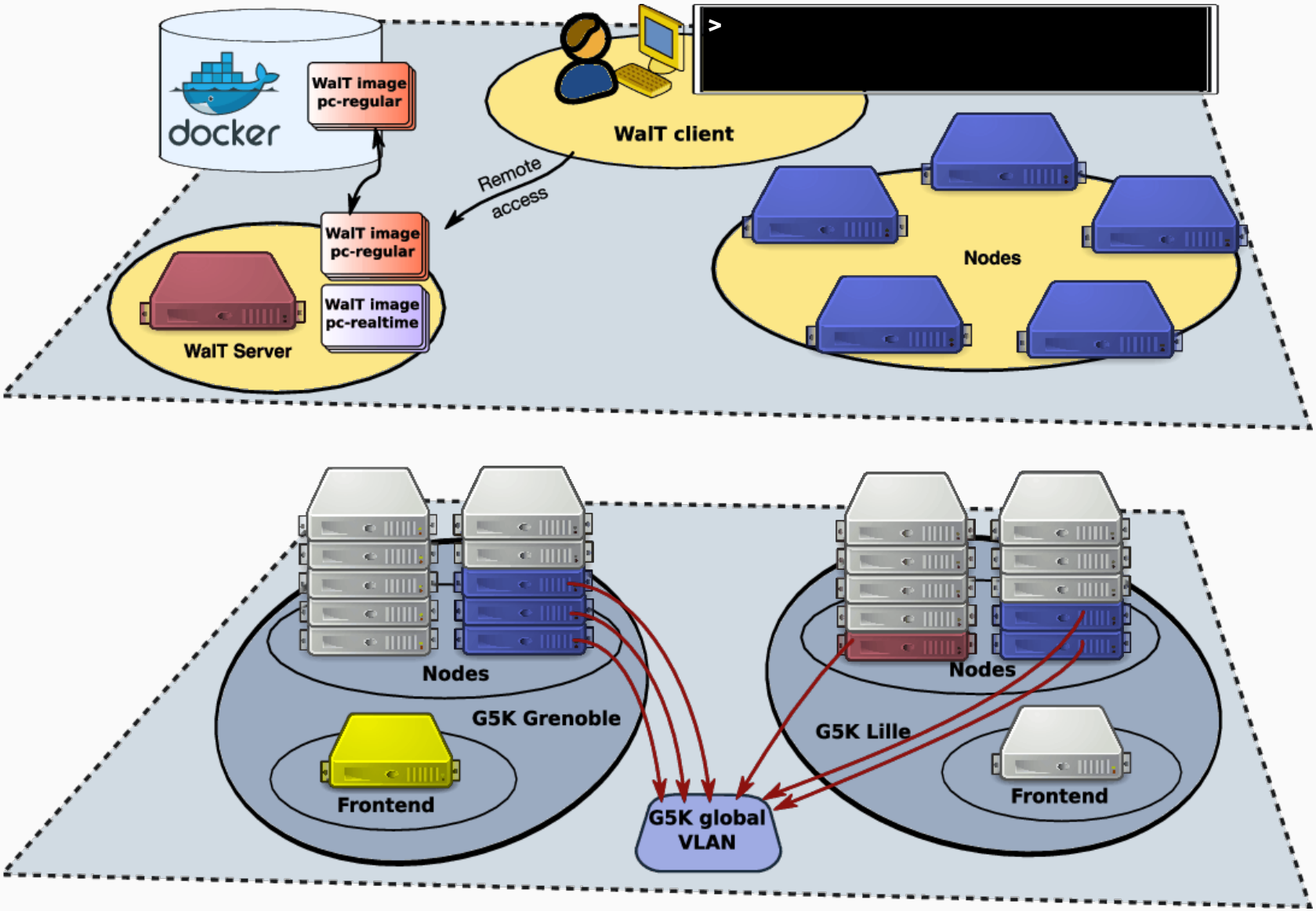
Sample use case



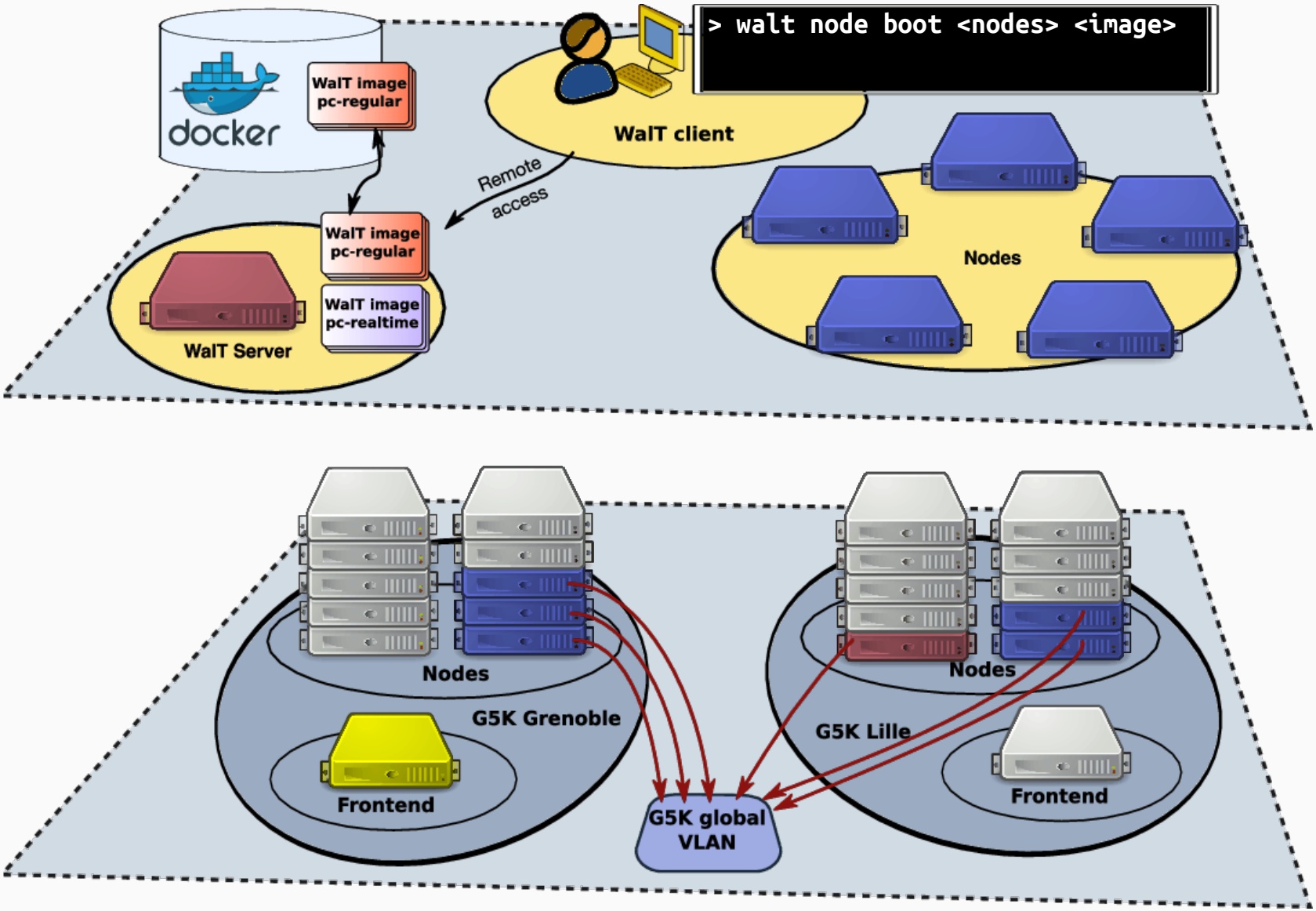
Sample use case



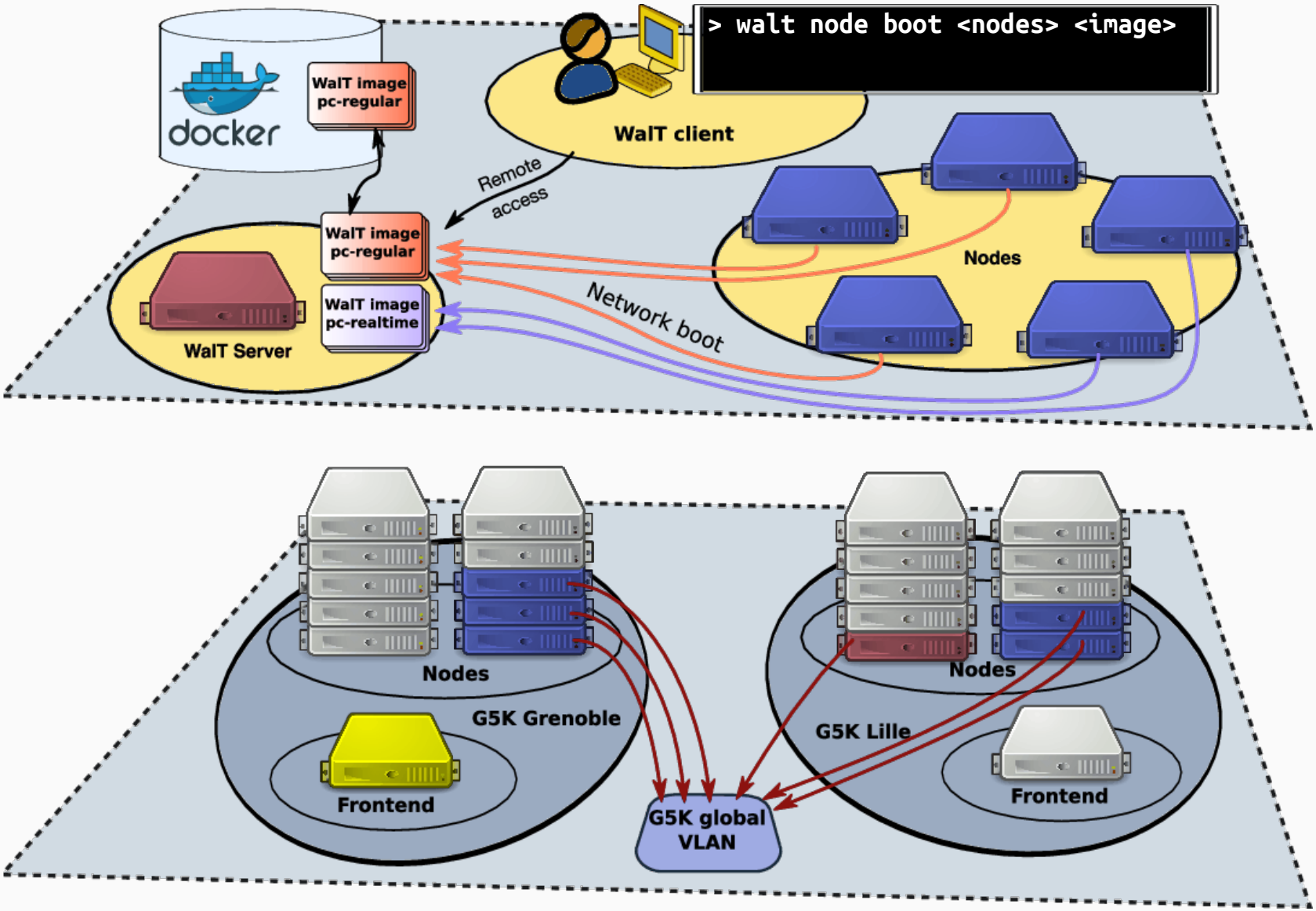
Sample use case



Sample use case



Sample use case



Agenda

Motivation

Sample use case

WalT & fast OS prototyping

OS deployment rework

Related news

Fast OS prototyping using WalT

WalT allows fast OS prototyping:

Fast OS prototyping using WalT

WalT allows fast OS prototyping:

- You can **modify an OS image** very easily...
  - `walt image shell <image>` (among other commands)
  - ... and save your changes in a few seconds
    (relies on docker image layered approach)

Fast OS prototyping using WalT

WalT allows fast OS prototyping:

- You can **modify an OS image** very easily...
  - `walt image shell <image>` (among other commands)
  - ... and save your changes in a few seconds
    (relies on docker image layered approach)
- You can **deploy an OS image** on nodes quickly:
  - `walt node boot <nodes> <image>`
  - Nodes are usually booted in less than 1 min
    (on a local testbed with Raspberry Pi boards)

Fast OS prototyping using WalT

WalT allows fast OS prototyping:

- You can **modify an OS image** very easily...
  - `walt image shell <image>` (among other commands)
  - ... and save your changes in a few seconds
    (relies on docker image layered approach)
- You can **deploy an OS image** on nodes quickly:
  - `walt node boot <nodes> <image>`
  - Nodes are usually booted in less than 1 min
    (on a local testbed with Raspberry Pi boards)
- **Reproducibility** is ensured at each node reboot
  - Nodes are stateless
  - Changes made wrt. image files are discarded on reboot

Agenda

Motivation

Sample use case

WalT & fast OS prototyping

OS deployment rework

Related news

WalT node deployment

How is OS deployment handled by WalT?

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure

About network boot

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure

About network boot

- *OS image remains on walt server only*
  (it is never transfered as a whole to the node)

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure

About network boot

- *OS image remains on walt server only*
  (it is never transfered as a whole to the node)
- Server exposes content of OS image as an **NFS network share**

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure

About network boot

- *OS image remains on walt server only*
  (it is never transfered as a whole to the node)
- Server exposes content of OS image as an **NFS network share**
- When booted, the whole OS of node is seated on this NFS share

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure
- Steps, for a light debian OS:

Step	**Description**	**Delay A**
0	Server: Route requests to new image	fast
1	Node: Hardware reboot	≈2min10
2	Node: Network bootloader	fast
3	Node: Wait for OS boot	≈15s
	Total procedure	**≈2min30**

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure
- Steps, for a light debian OS:

Step	**Description**	**Delay A**	**Delay B**
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈4min
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈6min50**

- Delay B: walt node and walt server are at a different G5K site.

WalT node deployment

How is OS deployment handled by WalT?

- As a regular **network boot** procedure
- Steps, for a light debian OS:

Step	**Description**	**Delay A**	**Delay B**
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈4min
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈6min50**

- Delay B: walt node and walt server are at a different G5K site.

WalT node deployment – avoiding TFTP

- Goal: Improve the slow bootloader step in case B

WalT node deployment – avoiding TFTP

- Goal: Improve the slow bootloader step in case B
- Diagnosis:
  - Kernel and initrd (≈30MB) downloaded from server site to node site

WalT node deployment – avoiding TFTP

- Goal: Improve the slow bootloader step in case B
- Diagnosis:
  - Kernel and initrd (≈30MB) downloaded from server site to node site
  - iPXE bootloader uses **TFTP protocol** for this

WalT node deployment – avoiding TFTP

- Goal: Improve the slow bootloader step in case B
- Diagnosis:
  - Kernel and initrd (≈30MB) downloaded from server site to node site
  - iPXE bootloader uses **TFTP protocol** for this
  - Transfers need ≈**30 000 round-trips** between sites to complete!

WalT node deployment – avoiding TFTP

- Goal: Improve the slow bootloader step in case B
- Diagnosis:
  - Kernel and initrd (≈30MB) downloaded from server site to node site
  - iPXE bootloader uses **TFTP protocol** for this
  - Transfers need ≈**30 000 round-trips** between sites to complete!
- Solution: let iPXE **use HTTP instead of TFTP**.

WalT node deployment – avoiding TFTP

- New numbers:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈35s
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈3min25**

WalT node deployment – avoiding TFTP

- New numbers:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈35s
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈3min25**

- Result: **this delay was reduced from ≈4min to ≈35s**.

WalT node deployment – avoiding TFTP

- New numbers:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈35s
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈3min25**

- Result: **this delay was reduced from ≈4min to ≈35s**.
- Analysis: better, but such a transfer between sites should be <1s. (Bootloader network driver is basic and suboptimal)

WalT node deployment – avoiding TFTP

- New numbers:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node: Hardware reboot	≈2min10	≈2min10
2	Node: Network bootloader	fast	≈35s
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈2min30**	**≈3min25**

- Result: **this delay was reduced from ≈4min to ≈35s**.
- Analysis: better, but such a transfer between sites should be <1s. (Bootloader network driver is basic and suboptimal)

WalT node deployment – avoiding hardware reboot

- Goal: Avoid hardware reboot

WalT node deployment – avoiding hardware reboot

- Goal: Avoid hardware reboot
- Idea: **Kexec** technique can be used to directly boot a new kernel (bypassing hardware reboot)

WalT node deployment – avoiding hardware reboot

- Goal: Avoid hardware reboot
- Idea: **Kexec** technique can be used to directly boot a new kernel (bypassing hardware reboot)
- Solution:
  - Implement this kexec-boot in `[image]:/bin/walt-reboot`

WalT node deployment – avoiding hardware reboot

- Goal: Avoid hardware reboot
- Idea: **Kexec** technique can be used to directly boot a new kernel (bypassing hardware reboot)
- Solution:
  - Implement this kexec-boot in `[image]:/bin/walt-reboot`
  - Mimic network bootloader behaviour:
    download and boot the kernel of the new image
    (not the current one!)

WalT node deployment – avoiding hardware reboot

- We now have different steps:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	<1s
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈20s**	**≈40s**

WalT node deployment – avoiding hardware reboot

- We now have different steps:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	<1s
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈20s**	**≈40s**

- Results:
  - **hardware reboot delay completely eliminated**.

WalT node deployment – avoiding hardware reboot

- We now have different steps:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	$<$1s
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈20s**	**≈40s**

- Results:
  - **hardware reboot delay completely eliminated**.
  - **kernel & initrd** now transfered by `walt-reboot` (was iPXE)

WalT node deployment – avoiding hardware reboot

- We now have different steps:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	<1s
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈20s**	**≈40s**

- Results:
  - **hardware reboot delay completely eliminated**.
  - **kernel & initrd** now transfered by `walt-reboot` (was iPXE)
  - Acceptable delay for the whole procedure.

WalT node deployment – avoiding hardware reboot

- We now have different steps:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	<1s
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈35s
	Total procedure	**≈20s**	**≈40s**

- Results:
  - **hardware reboot delay completely eliminated**.
  - **kernel & initrd** now transfered by `walt-reboot` (was iPXE)
  - Acceptable delay for the whole procedure.

WalT node deployment – avoiding NFS

- Goal: Reduce OS boot delay in case B

WalT node deployment – avoiding NFS

- Goal: Reduce OS boot delay in case B
- Diagnosis: **NFS is slow when client-server latency is high**.

WalT node deployment – avoiding NFS

- Goal: Reduce OS boot delay in case B
- Diagnosis: **NFS is slow when client-server latency is high**.
- Solution: use **NBFS** instead.

WalT node deployment – avoiding NFS

- Goal: Reduce OS boot delay in case B
- Diagnosis: **NFS is slow when client-server latency is high**.
- Solution: use **NBFS** instead.

About NBFS

- An experimental **network filesystem** I am working on.
- Specialized for **network booting**.
- Uses **speculation**: good performance even when **latency** is high.
- Improves responsiveness after bootup too (e.g. reduced ssh login delay).
- A research paper is being written with R.Lachaize, F.Rousseau, A.Duda.

WalT node deployment – avoiding NFS

- New numbers with NBFS instead of NFS:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	fast
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈15s
	Total procedure	**≈20s**	**≈20s**

WalT node deployment – avoiding NFS

- New numbers with NBFS instead of NFS:

Step	Description	Delay A	Delay B
0	Server: Route requests to new image	fast	fast
1	Node (walt-reboot): get kernel & initrd	fast	fast
2	Node (walt-reboot): kexec new kernel	fast	fast
3	Node: Wait for OS boot	≈15s	≈15s
	Total procedure	**≈20s**	**≈20s**

- Result: **similar delay for remote and local bootup**.

Agenda

Motivation

Sample use case

WalT & fast OS prototyping

OS deployment rework

Related news

Related news

- **NBFS** is still **experimental**
- **WalT-on-G5K** feature planning:
  - available with **WALT version 8** (end of june)
  - **NBFS** will **not** be included
  - we are discussing with G5K team for improvements / documentation etc.
- G5K team is working on improving `kadeploy` with **kexec** too (near future)

More info:

WalT-on-G5K demo:	https://vu.fr/walt-on-g5k
WalT website:	https://vu.fr/walt
Questions, WalT training requests:	etienne.duble@imag.fr