Going Full IPv6 in Kubernetes:

No Limits, Just 128 bits!

whoami

Ole Mathias Heggem

Redpill Linpro
Norway

- First Golden Kubestronaut in Norway
  - All CNCF certifications
- Loves all kind of tech, from microcontrollers to rockets
- Used IPv6 at home since 2013-ish
- Have my *own* ASN and a /29 block with IPv6 (and a /24 with IPv4)

GOLDEN
Kubestronaut

Redpill Linpro

Legacy IP (IPv4)

- Core networking protocol in today's internet
- Standardized in RFC 791 - **1981**
- Each packet contains a sender and a receiver address
  - Dot-decimal notation, example **10.13.37.1**
  - **32-bit**
  - Problem: $\mathbf{2^{32}}$ is only **4 294 967 296** addresses
    - Less than the current world population
- Workaround is Source Network Address Translation (NAT / SNAT)
  - It can “hide” multiple users behind a single IP
  - But brings complexity and has limitations

Why IPv6 now?

- Cloud Providers like AWS is (2023) charging for public IPv4 addresses
- Many ISPs are resorting to Carrier-Grade NAT (CGNAT) for its end users
  - Multiple layers of NAT = more complexity and issues
  - A lot of end-users behind same external IP
- Services can have rate-limit per IP
  - docker.io: 100 pulls per IPv4 address or IPv6 /64 subnet

RIPE IPv4 waiting list: https://www.ripe.net/manage-ips-and-asns/ipv4/ipv4-waiting-list/

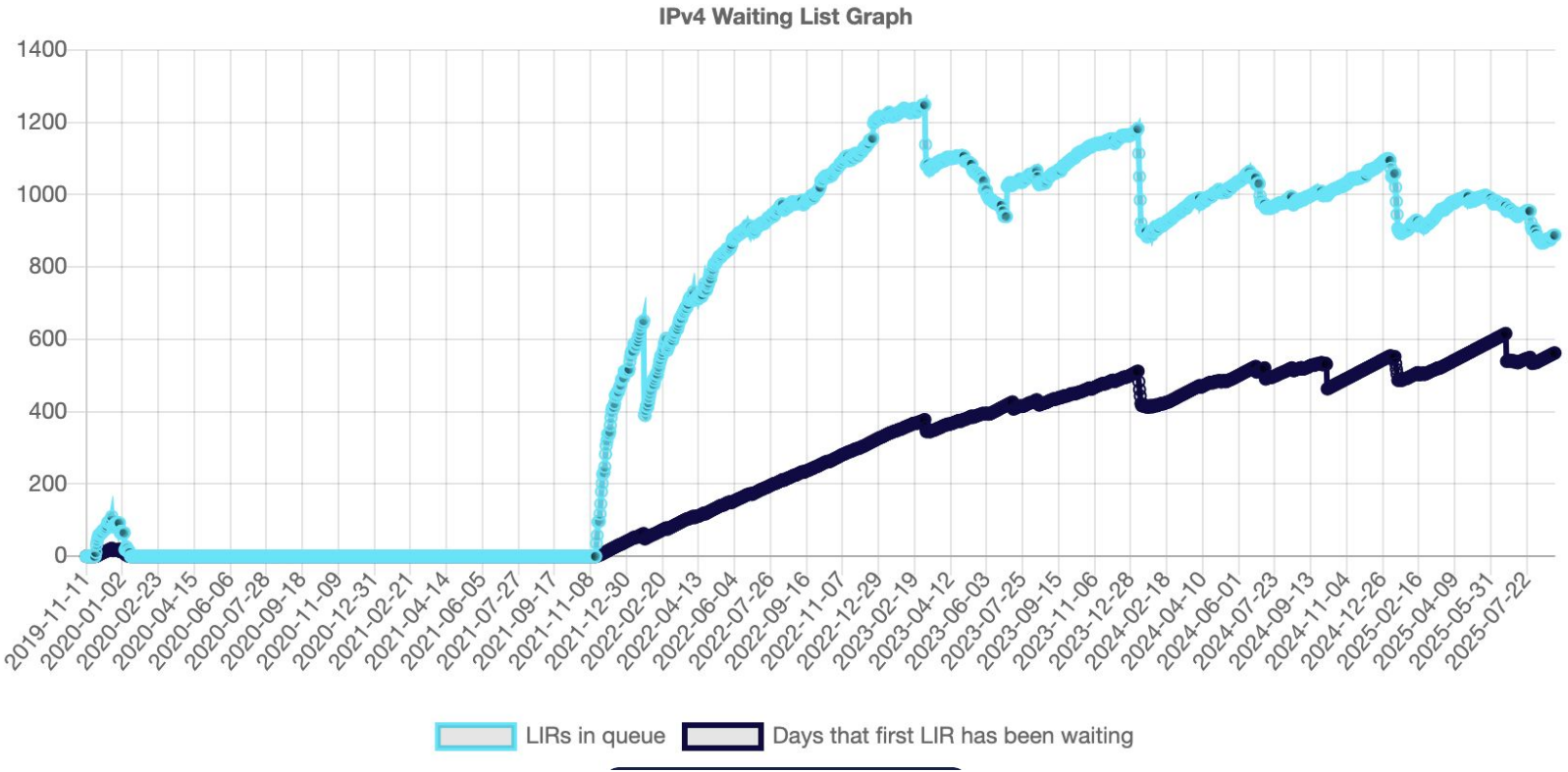
Why IPv6 now?

IPv4 Waiting List

LIRs in queue	888
Days that first LIR in queue has been waiting	562

RIPE IPv4 waiting list: https://www.ripe.net/manage-ips-and-asns/ipv4/ipv4-waiting-list/

Why IPv6 now?



The fix

- IPv6
- First draft in RFC 2460 - **1998**
- Larger address space
  - 128-bit

4 294 967 296

8 273 033 546

340 282 366 920 938 463 463 374 607 431 768 211 456

IPv6

IPv4

https://blog.apnic.net/2024/12/20/visualizing-the-scale-differences-of-ipv4-and-ipv6/

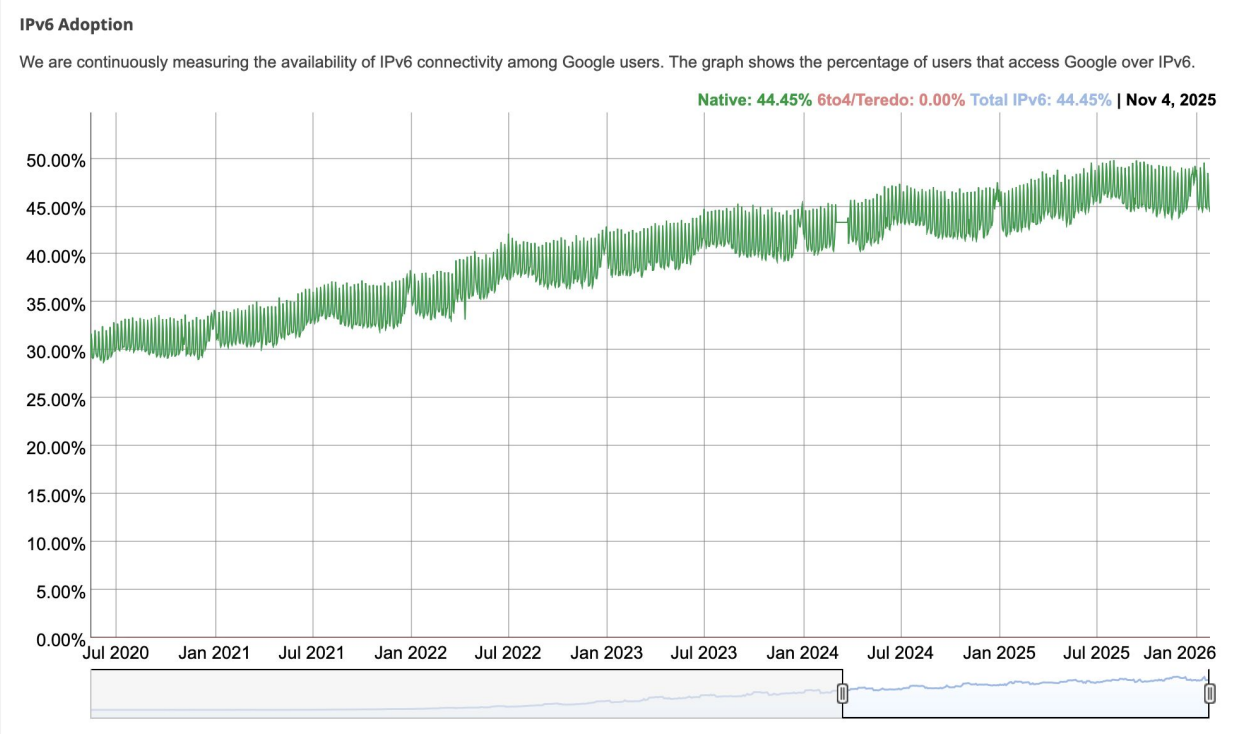
IPv6

- First draft in RFC 2460 - **1998**
- Larger address space - IPv6
  - 128-bit
  - **2^{128} = 340 282 366 920 938 463 463 374 607 431 768 211 456**
  - Hexadecimal notation - 8 groups of 16 bits each
    - 2a02:d140:c012:0003:0000:0000:0000:0122
    - 2a02:d140:c012:3::122
    - 2606:4700:4700::1111
- No need for NAT to extend address space

IPv6 status

Google is close to 50% adaptation

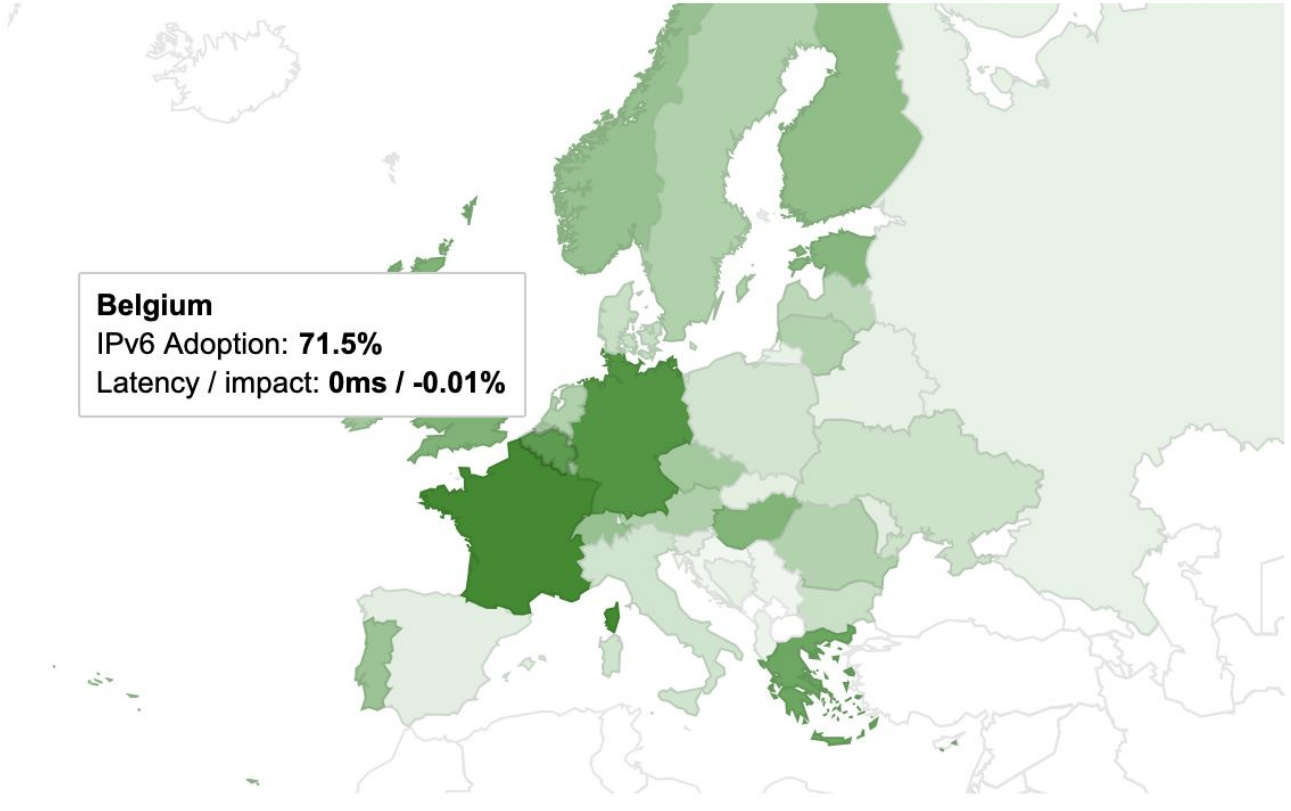
So end-users are getting there



Source: https://www.google.com/intl/en/ipv6/statistics.html

IPv6 Adoption | **Per-Country IPv6 adoption**

Per-Country IPv6 adoption



Source: https://www.google.com/intl/en/ipv6/statistics.html

Source: https://whynoipv6.com

Dual-Stack

- Dual Stack enabled hosts supporting both protocols
  - They have both IPv4 and IPv6 addresses
  - Can use either IPv4 / IPv6 if supported by the destination
    - Need IPv4 and/or IPv6 in DNS records
  - Modern operating systems prefer IPv6 over IPv4, but not guaranteed
- Adds complexity
  - You now have two stacks
  - Need to secure both stacks
  - Troubleshooting can be harder

Dual-Stack ?

- For clients this is mostly still needed (for now)
- For services exposed to end-users, this is still needed
- For servers that mostly talk to each other, why have two stacks?

IPv6 support in Kubernetes

- IPv6-only Clusters
  - Since K8s 1.9
  - Pods & services only have IPv6 addresses
- Dual-Stack Kubernetes clusters
  - GA since K8s 1.23
  - Every pod has an IP address from each family
  - K8s services can be declared as single-stack or dual-stack
- Each pod can have a public IPv6 Address
  - Like in IPv4 you will need to think about security
  - No need for NAT

K8s IPv6 support in Cloud Providers

- AWS EKS: Dual-stack
  - Each VPC get a /56
- GCP GKE: Dual-stack
- Azure AKS: Dual-stack
  - Uses NAT66 for IPv6
- Expect mixed maturity across providers

Public or Private Prefix

- Public:
  - Needs to be allocated from service provider
  - Probably needs BGP to route to your cluster
  - No NAT!
- Private:
  - Unique Local Addresses
    - Like RFC1918 in IPv4
  - Need NAT66 to reach the internet

Enabling IPv6 with a public prefix

- Request/allocate IPv6 prefix from ISP or cloud provider
- Example:
  - Service: /108
  - Pod subnets: /60
    - Each node gets its own /64
    - /60 you can have max 16 nodes
- Route the subnets to the nodes
  - BGP
- Security: “open” by default
  - Must harden access controls with ACLs
    - You should do this with IPv4 too

Enabling IPv6 with a private prefix

- Useful for isolated clusters (local, labs, dev, CI/CD)
  - Avoid dependency on public IPv6 allocations
- Use of Unique Local Addresses (ULA): fc00::/7
  - Like RFC1918 in IPv4
- No global reachability
  - Must use NAT66 or proxies for external access

Enabling Dual-Stack

- Nodes needs IPv4 and IPv6 addresses
  - Must be supported by the cloud platform
- kube-apiserver
  - –service-cluster-ip-range=<IPv4 CIDR>,<IPv6 CIDR>
- kube-controller-manager
  - –service-cluster-ip-range=<IPv4 CIDR>,<IPv6 CIDR>
  - –cluster-cidr=<IPv4 CIDR>,<IPv6 CIDR>
  - –node-cidr-mask-size-ipv4 & –node-cidr-mask-size-ipv6
- kube-proxy
  - –cluster-cidr=<IPv4 CIDR>,<IPv6 CIDR>
- kubelet
  - –node-ip=<IPv4>,<IPv6>
- CNI needs support and to be configured

Kubernetes CNI (Container Network Interface)

- Standard for container networking in Kubernetes
- Defines how Pods get IP addresses and connectivity
- Determines IPv6 feature availability in clusters

Cilium for IPv6?

- Supports IPv6-only and dual-stack clusters
- IPv6-aware load balancing (ClusterIP, NodePort, External)
- Hubble provides flow visibility, including IPv6 traffic

Dual-Stack Kubernets Services

- Since K8s 1.21, each service spec has new fields:
- *.spec.ipFamilies*:
  - [“IPv4”]
  - [“IPv6”]
  - [“IPv4”, “IPv4”]
- *.spec.ipFamilyPolicy*:
  - SingleStack (default)
  - PreferDualStack
  - RequireDualStack

Exploring IPv6-Only Deployments

- Escape IPv4 & NAT complexity
  - With a public IPv6 prefix
- Simplify addressing: every Pod gets a globally unique IP
- End-to-end connectivity without translation layers (to other IPv6 services)

Enabling IPv6-only in Cilium

- From Cilium 1.18 IPv6 underlay is supported

```
❯ helm install cilium cilium/cilium -n kube-system \
      --version 1.18.0 \
      --set ipv6.enabled=true \
      --set ipv4.enabled=false \
      --set underlayProtocol=ipv6
```

K3s & Cilium IPv6-only

```
curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC='--flannel-backend=none --disable-network-policy
--cluster-cidr=fd77:19f3:99c2::/60 --service-cidr=fd96:4a16:c0ea::/112' sh -
```

Cilium Helm Values

```yaml
ipam:
  mode: kubernetes
operator:
  replicas: 1
ipv4:
  enabled: false
ipv6:
  enabled: true
underlayProtocol: ipv6
enableIPv4Masquerade: false
enableIPv6Masquerade: true
kubeProxyReplacement: false
```

```
root@ole-k3s-ipv6:~# kubectl get pods -o wide -A
NAMESPACE     NAME                                      READY   STATUS      RESTARTS   AGE   IP                          NODE           NOMINATED NODE   READINESS GATES
cilium        cilium-bcx8c                              1/1     Running     0          12m   2a12:6bc0:1337:100::138     ole-k3s-ipv6   <none>           <none>
cilium        cilium-envoy-qf46f                        1/1     Running     0          12m   2a12:6bc0:1337:100::138     ole-k3s-ipv6   <none>           <none>
cilium        cilium-operator-6ff676d75f-k59tc          1/1     Running     0          12m   2a12:6bc0:1337:100::138     ole-k3s-ipv6   <none>           <none>
kube-system   coredns-7f496c8d7d-6bvxd                  1/1     Running     0          19m   fd77:19f3:99c2::4101        ole-k3s-ipv6   <none>           <none>
kube-system   helm-install-traefik-c57k5                0/1     Completed   2          19m   fd77:19f3:99c2::db86        ole-k3s-ipv6   <none>           <none>
kube-system   helm-install-traefik-crd-7x867            0/1     Completed   0          19m   fd77:19f3:99c2::cba9        ole-k3s-ipv6   <none>           <none>
kube-system   local-path-provisioner-578895bd58-m7gwr   1/1     Running     0          19m   fd77:19f3:99c2::b227        ole-k3s-ipv6   <none>           <none>
kube-system   metrics-server-7b9c9c4b9c-75cv6           1/1     Running     0          19m   fd77:19f3:99c2::108e        ole-k3s-ipv6   <none>           <none>
kube-system   svclb-traefik-50b11e60-4t5kn              2/2     Running     0          11m   fd77:19f3:99c2::d19c        ole-k3s-ipv6   <none>           <none>
kube-system   traefik-6f5f87584-9m82k                   1/1     Running     0          11m   fd77:19f3:99c2::2119        ole-k3s-ipv6   <none>           <none>


root@ole-k3s-ipv6:~# kubectl get svc -A
NAMESPACE     NAME             TYPE           CLUSTER-IP              EXTERNAL-IP               PORT(S)                      AGE
cilium        cilium-envoy     ClusterIP      None                    <none>                    9964/TCP                     13m
cilium        hubble-peer      ClusterIP      fd96:4a16:c0ea::7d80    <none>                    443/TCP                      13m
default       kubernetes       ClusterIP      fd96:4a16:c0ea::1       <none>                    443/TCP                      20m
kube-system   kube-dns         ClusterIP      fd96:4a16:c0ea::a       <none>                    53/UDP,53/TCP,9153/TCP       20m
kube-system   metrics-server   ClusterIP      fd96:4a16:c0ea::6cf5    <none>                    443/TCP                      20m
kube-system   traefik          LoadBalancer   fd96:4a16:c0ea::d78e    2a12:6bc0:1337:100::138   80:31047/TCP,443:31744/TCP   12m
```

Talking to IPv4-only services from a IPv6-only cluster

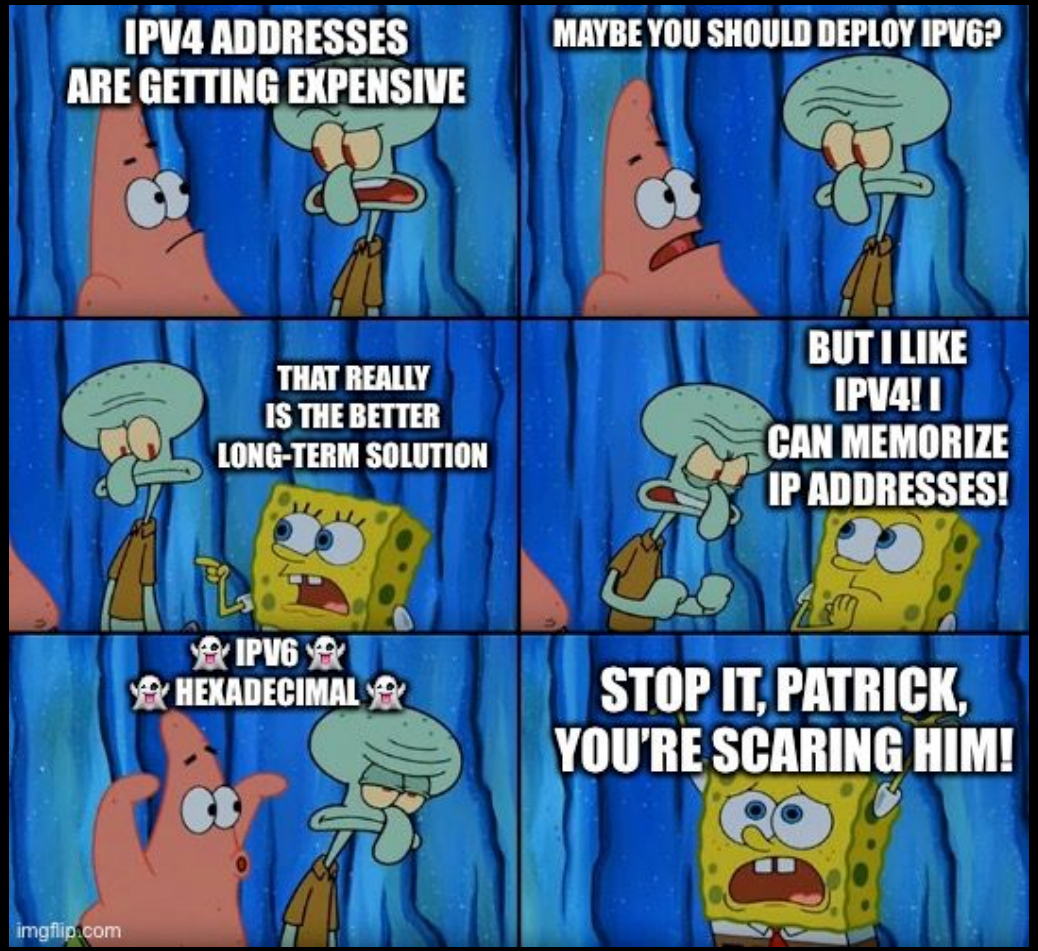
- Use a web proxy or a registry as a pull through cache
- NAT64
  - Translation mechanism: IPv6 -> IPv4
  - Allows IPv6-only clients (Pods) to access IPv4 resources
  - Works together with DNS64 or CLAT for hostname resolution

Challenges

- Getting the rest of the organization onboard
- For many IPv6 is still new and unknown

Challenges

- Many apps & libraries still assume a IPv4 address
- Some managed services (LBs, storage, monitoring) only support IPv4
- Tooling may lag behind in IPv6 support


IPV4 ADDRESSES ARE GETTING EXPENSIVE
MAYBE YOU SHOULD DEPLOY IPV6?
THAT REALLY IS THE BETTER LONG-TERM SOLUTION
BUT I LIKE IPV4! I CAN MEMORIZE IP ADDRESSES!
IPV6
HEXADECIMAL
STOP IT, PATRICK, YOU'RE SCARING HIM!
imgflip.com