# AKADEMIA KUBERNETESA

## IPVG | SECURITY

Vladimir 'vovcia' Mitiouchev

19.03.2021 / 18:00

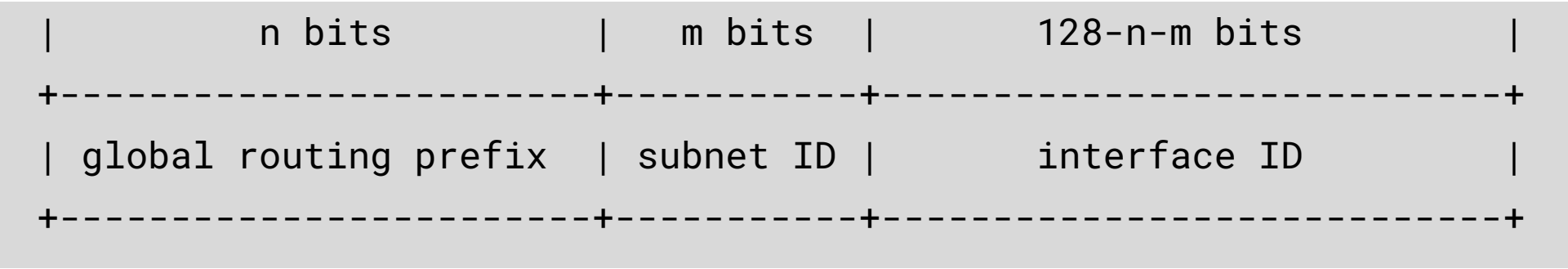
# Prerequisites

- Linux machine ready for Kubernetes:
  - any compatible CRI: `containerd`, `docker`, `cri-o*`
  - `kubeadm`, `kubelet`, `kubectl >= 1.20`
  - `helm`
- Global IPv6 (min. `/64` or `/128`)

*using cri-o is somewhat painful

# Global Unicast Address

# Global Unicast Address



# IPv6 address plan


- Linux server: `/64`
- Home NAS: `/64`
- IoT subnet: `/64`
- Media subnet: `/64`
- WiFi subnet: `/64`
- Work subnet: `/64`
- Router subnet: `/64`
- Secure subnet: `/64`
- Home prefix: `/56` = 256 subnets = 16 groups by 16 subnets
- Business prefix: `/48` = 65536 subnets = 256 groups by 256 subnets
- ISP allocation: `/32` = 16777216 home users or 65536 business users

# IPv6 address notation

- 128 bits are represented as **8** groups of **4** hexadecimal numbers separated by colon
- In any group leading `0` can be omitted
- Sequence of `0:` can be replaced by `::`

```
2001:db8:0:0:1:0:0:1

2001:0db8:0:0:1:0:0:1

2001:db8::1:0:0:1

2001:db8::0:1:0:0:1

2001:0db8::1:0:0:1

2001:db8:0:0:1::1

2001:db8:0000:0:1::1

2001:DB8:0:0:1::1

All of the above examples represent the same IPv6 address.
```

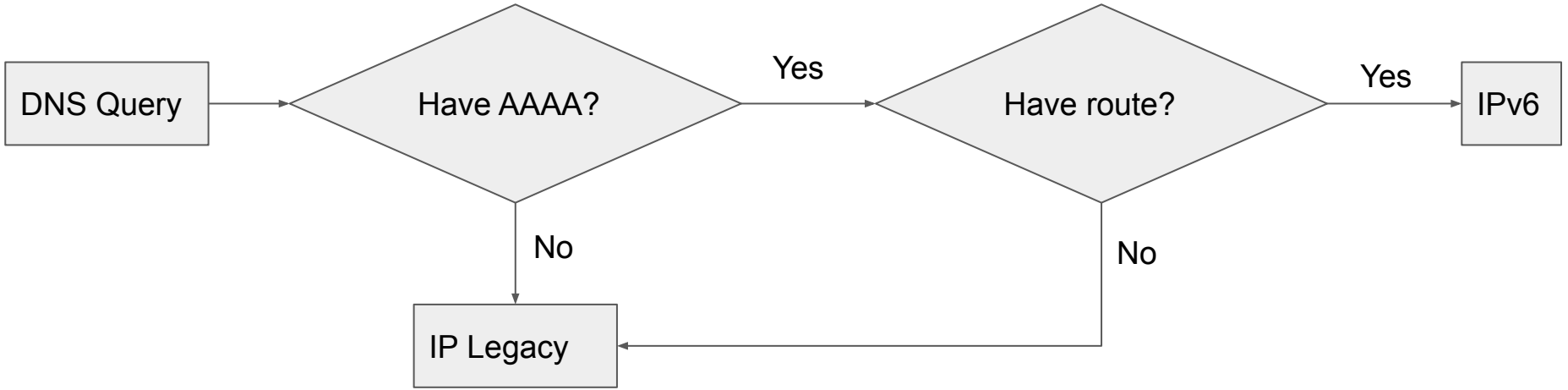
# Special addresses in IPv6*

| | |
|---|---|
| Unspecified | `::` |
| Loopback | `::1/128` |
| NAT64 | `64:ff9b::/96` |
| Documentation | `2001:db8::/32` |
| **Unique local addresses** | **`fc00::/7`** |
| Link-local | `fe80::/10` |
| Multicast | `ff00::/8` |

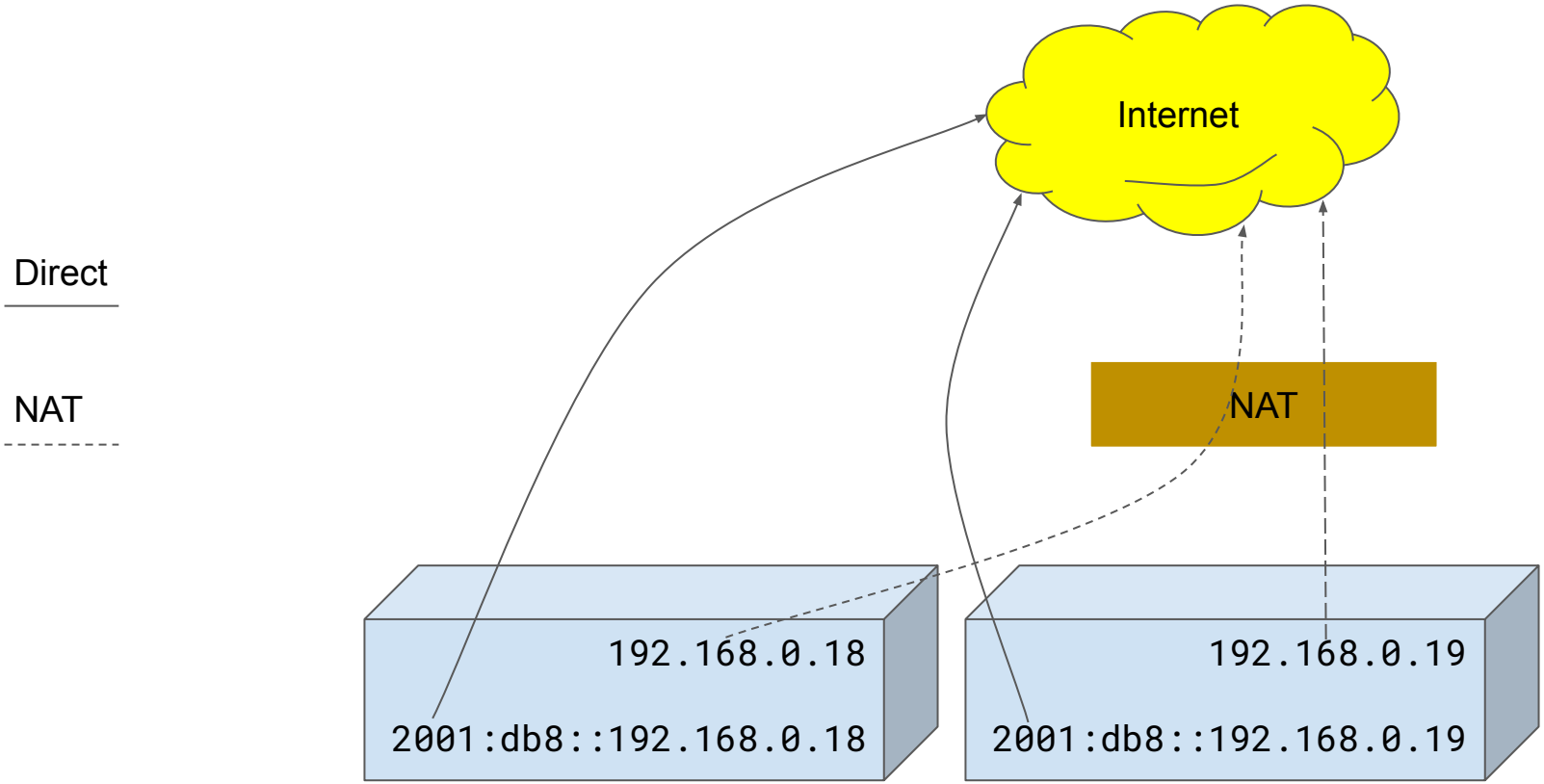
*this is not full list

# IPv6 transition mechanisms: Dual Stack

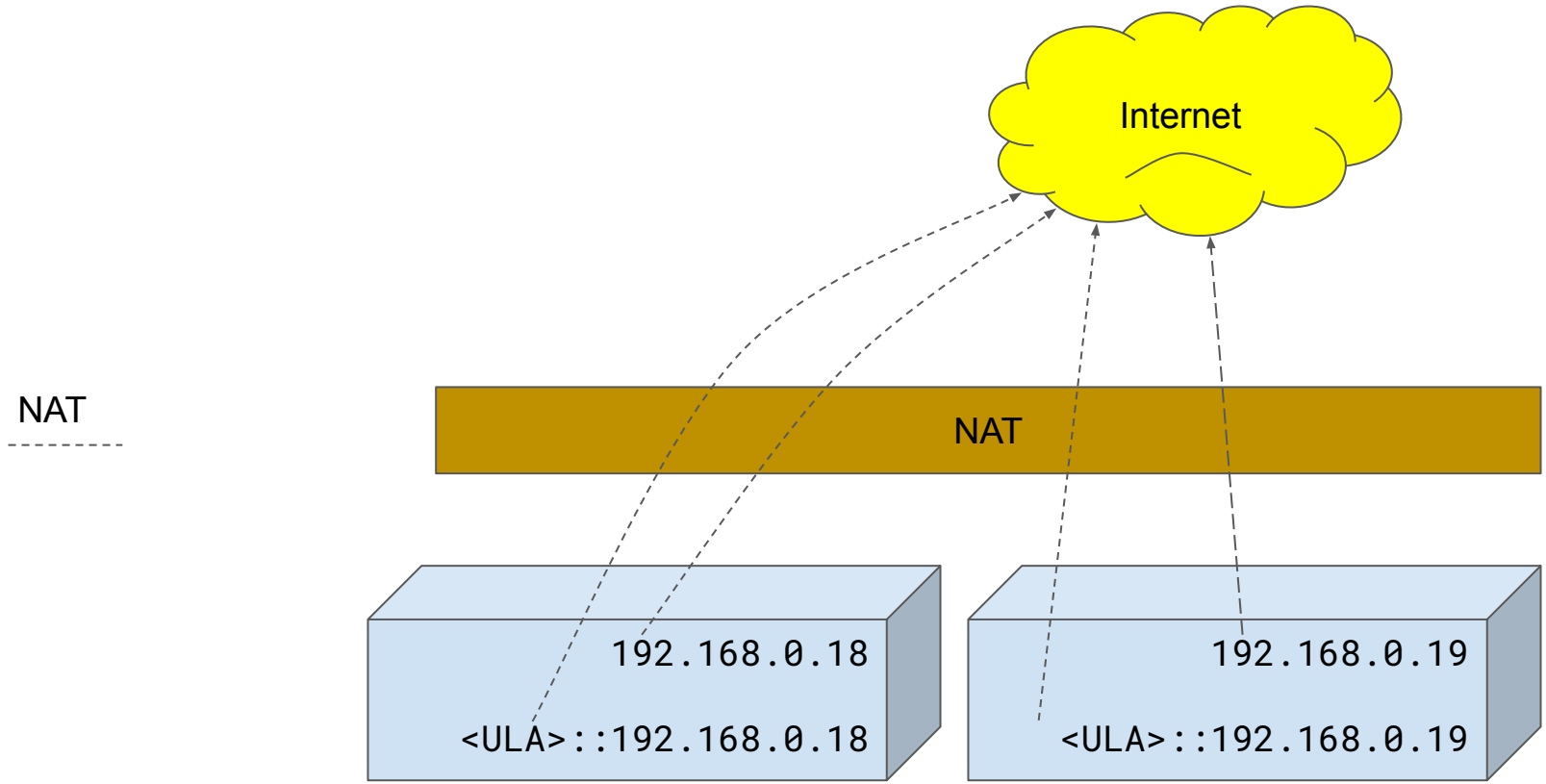
- Simultaneous IP Legacy and IPv6 addresses are configured
- Applications supports both protocols
- Applications prefers IPv6 if available



# Dual Stack with Global Prefix



# Dual Stack with ULA



# Install demo

```
$ git clone https://git.vovcia.net/clug/k8s-ipv6-demo
$ vim k8s-ipv6-demo/install.sh
$ sudo sh k8s-ipv6-demo/install.sh
```

# Kubernetes NetworkPolicy to the rescue!!

- Implemented by network plugin
- Default allow if no policy attached
- Default deny if any NetworkPolicy in a namespace is selected
- Network Policies are additive
- Egress policy on the source and the ingress policy on the destination need to allow the traffic

# NetworkPolicy editor

https://cilium.io/blog/2021/02/10/network-policy-editor

https://editor.cilium.io/

# NetworkPolicy common pitfalls

- Mistake 1: Not Using a Namespace Selector
- Mistake 2: "There is no way it’s DNS..."
- Mistake 3: Using Traditional Networking Constructs
- Mistake 4: Misunderstanding How Policy Rules Combine
- Mistake 5: Confusing Different Uses for “{}”

# IP Spoof Prevention

https://cilium.io/blog/2020/06/29/cilium-kubernetes-cni-vulnerability

# Segment Routing

# Contiv-SRv6

https://archive.fosdem.org/2020/schedule/event/rethinking_kubernetes_networking_with_srv6/attachments/slides/3687/export/events/attachments/rethinking_kubernetes_networking_with_srv6/slides/3687/20200203_FOSDEM_SRv6_K8s.pdf

https://github.com/netgroup/SRv6-net-prog

https://www.segment-routing.net/tutorials/2017-12-05-srv6-introduction/

# Thank you!

Support CLUG for moar servers and cool events! https://zrzutka.pl/325rh5

Slides and code - https://git.vovcia.net/clug/k8s-ipv6-demo

# IPv6 transition mechanisms*

- SIIT-DC — *Stateless IP/ICMP Translation for IPv6 Data Center Environments*
- NAT64+DNS64 — *Stateful NAT64 Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*
- 464XLAT — *Combination of Stateful and Stateless Translation*

**See https://www.jool.mx/en/intro-xlat.html for more details**

*this is not full list

# NAT64 + DNS64