PLAYING OPENSTACK "JENGA"

About me

- Hi, I’m Jonas!
- I have weird cravings to self-host *everything*…
- I’ve been playing with Linux for over 10 years, with my first server in production being deployed in 2012 (it was recently decomissioned in 2024)
- Ex-Nasdaq application support, software developer and infosec specialist. Currently have my own little company and do various weird things!

So what are we talking about?

I want to share a little bit about how I ended up running my own Openstack cluster for lab environment purposes.

I'll cover:

- The general idea
- The project target
- The implementation attempts (there's more than a few...)
- And finally the eventual success!

What's the idea?

- **We help teach a few courses, one of them being about Virtualization and Cloud**
- **We use to get sponsored by various cloud providers here in Sweden (shoutout to Elastx and Cleura! <3)**
- **The problem is that students sometimes mess up... 30-40 free accounts even with minimal compute adds up quickly!**
- **And all of that sometimes leads to unintentional abuse:**
  - Credential leaks
  - (Un)intentional crypto miners
  - Support abuse

What’s the idea?

- **It’s unsustainable long-term… so during one lunch we started to think: There’s plenty of pretty affordable used servers you can rent on OVH or Hetzner, why not run it ourselves?**

  HOW HARD CAN IT BE?!

  At this moment… The future still looked so bright.

The project target

- **Stable, simple and disposable**
  - This means we can essentially focus on purely compute, so we'll need CPU, RAM and some Storage
- **Maintainable by a single person**
  - We're tiny… there's very little resources and we need to make sure we can support it.
- **Can't be outrageously expensive**
  - Budgets are often tight, so targeting around 10 EUR per student would be preferable.

The project target – Constraints

- **Needs to be able to run on any provider**
  - Regardless if it’s on-prem or renting from a provider, the deployment should be almost the same
- **No local network between nodes**
  - Each provider is different, some offer it, some don’t. Being able to mix providers might be beneficial as well!
- **Reasonably secure and performant**
  - Students are allowed and will make mistakes, figuring out a way to protect them is paramount to both them and us. (If some kind of malicious things happen the server providers will probably not be happy…)
- **Can’t have complicated hacks**
  - The platform is going to be used by students, often new to IT overall. So the setup and usage should be as simple as any other Openstack cloud (or at least as close to it)

Attempt 1 – Wow… is it really this simple?

- **I've heard about Canonical's MicroStack**
- **Personally I'm not a fan of how Canonical wants to do things and how it does it in Ubuntu (mainly snap)**
- **But it looked… kind of amazing!**
  - *"It installs in minutes through a friendly interface, making OpenStack fully accessible to those with no previous experience."*

    It sounds exactly like me…

Attempt 1 – Hah… it’s not that easy.

- **I gave it a shot… and had a horrible experience.**
- **The architecture is quite complicated:**
  - It starts with node preparation, that installs “Juju”, Canonical’s orchestration tool
  - Sunbeam, the Microstack deployment tool uses “Juju” to install MicroK8S and MicroCeph snap packages
  - Joins all the nodes in a cluster for both Ceph and Kubernetes.
  - Adds the Kubernetes cluster into Juju
  - And finally deploys all the Openstack components on top Kubernetes using Juju.
  - All of the components are Canonical provided and configuration is mostly handled through Juju.

Attempt 1 – Good and bad

- **The good:**
  - It definitely is quite easy, just following the Sunbeam interactive setup and copy pasting some commands
  - It’s very opinionated, so if you don’t know anything, they do it for you!
- **The bad:**
  - I don’t have a local network, so I need to do this over the internet or using some kind of VPN. Kubernetes and other tools often struggle with that a little bit performance wise.
  - It’s very complicated… if you don’t have Ceph or K8s experience, modifying, fixing or even maintining the setup is close to impossible.
  - It’s opinionated – this is also bad, because if things go wrong (and they do) it’s hard to understand how everything works and even then difficult to make small fixes since everything follows this very strict structure.

Attempt 1 – Final thoughts

- Maybe in a simple local cluster or even a single node, if you don’t know anything, this would work great.
- I think it’s very easy to grow out of and when you do… the opinionated decisions hurt a lot
- Finally the stack is intricate… lots of moving pieces and without having experience in each of them, there’s going to be a lot of time sunk on just figuring out how to do maintenance.

Attempt 2 – Openstack-Ansible

- **Ok… So the simple super duper easy installers don’t work for me, let’s try something a bit more sensible, flexible and battle-tested.**
- **As the name suggests, it’s a suite of scripts and ansible roles/playbooks to setup Openstack**
  - Based on LXC
  - Less overhead and much simpler architecture
  - For storage, there’s many options, one of them LVM!

Attempt 2 – How did it go?

- **This was actually functioning!**
  - The documentation is bit hard to follow and it took a few tries to figure out how the configuration file needs to look and what “knobs” are operated, but after trying a couple of things it becomes much easier.
  - With the simpler architecture it was much easier to debug and figure out problems and implement temporary/permanent solutions
  - And it was highly available! 3 nodes, each ran the entire control plane, all automated through Openstack-Ansible

Attempt 2 – But I was stupid and greedy

- **Most of these solutions need and expect an L2 local network, both for container-container communication and the virtual networks users create.**
  - While we didn’t have one and most VPNs are L3!
  - I setup GRETAP tunnels between the nodes with a bridge in the “middle” node to have a “dumb” L2 network. And it worked… but it was jank-city!
  - And I was greedy, since I had 3 nodes… I thought “Hey… Why not have HA?”

Attempt 2 – The good and the bad

- **The good:**
  - It was finally functional and even highly available
  - The setup was clear and architecture simple
  - With clear and easy opportunities to modify based on our needs
- **The bad:**
  - My “jank-city” network couldn’t support the highly available control plane… so it would lead to random cluster disconnects for the database and the messaging queues, leading to an almost constant cascading (disconnect-reconnect-recover cluster) effect, slowing the environment down to almost inoperable.
  - And these issues would pop up mostly with load, a few VMs would be fine, but a few users caused issues quickly…

Attempt 2b – Maybe I can fix it?

- **While not quickly… it became clear that my “jank-city” network is just not up to the task. The almost constant cluster rebalances basically make the platform unusable under almost all circumstances!**
- **So what can we do…:**
  - I need L2 and our server provider Hetzner could offer that by moving server closer together and connecting cables. But it incurs significant costs and the project is supposed to be very flexible and dynamic
  - But all of the components that are running are familiar to me… so maybe I can pull some other tricks from my Networking jank box?

Attempt 2b – FRR to the rescue

- **A while ago I was playing around with different VPN technologies and one was very interesting, because of the low overhead – NHRP**
  - If you've heard of or did something with Cisco DMVPN, NHRP is what that product is based on.
  - Basic concept of how it works, it's a hub and spoke model:
    - There's a central hub that all the clients (spokes) connect to initially
    - And then the central hub sends "shortcuts" telling nodes how to directly reach their destination, instead of going through the hub, achieving spoke-to-spoke communication
    - It's a dynamic VPN done with multipoint GRE tunnels, where the hub is connected to everyone and any initial communication can go through it, while it sends you shortcuts so that you can later communicate directly with the node!
- **It's really cool and FRR (Free Range Routing) package in Linux implements it!**

Attempt 2b – But is that L2…?

- **No it is not… NHRP like most VPNs handle only L3 connectivity**
- **However… since all of the components are much simpler and easier to modify so I:**
  - Adjusted all the LXC containers to use “routed” mode on it’s interfaces. This add a route for the containers IP address to the hosts routing table and vice versa within the container
  - Changed the IPs from /24 to /32 in all containers
  - And finally setup BGP on FRR to share the hosts routing table with other nodes in the VPN
- **With this… we have container to container communication all over L3, nicely routed!**
- **AND IT WORKS! *Ish*…**

Attempt 2b – The good and the bad

- **The good:**
  - This seemed to have stabilized the network, no more database or MQ rebuilds, everything while not fast… mostly stable!
  - Much less jank, no more GRETAP tunnels with bridges in between.
  - FRR with BGP could also handle VXLAN interfaces used for virtual networks created by the students.
- **The bad:**
  - But it didn’t work out… Had a weird bug of NHRP getting locked up after a while. Where it would just lose all connectivity to other nodes, including the hub. It was either a bug or most likely some configuration or maybe an edge case somewhere.
- **I still think this could probably work out… but never tried it again.**

Attempt 3 - 3rd time's the charm?

- **Oh god… please work!**
- **I liked Openstack-Ansible and all that it brings, but just need to figure out how to make the network solid.**
- **So, I simplified it:**
  - Using Wireguard I created a mesh VPN between all nodes, it serves as our underlay network
  - With FRR and VXLAN, I setup an overlay network between all the nodes for all the container/component communication. It's L2, so we don't need LXC hacks I did before
  - FRR also handles any virtual networks that are created by the students in Openstack
  - And… no more HA. It was never a target, it made everything so much more complicated and we didn't even need it!

Attempt 3 – Finally progress!

- **It works and it’s performant**
  - With no HA on the components, the architecture is simple, no more cluster rebuilds even if there are intermittent disconnects
  - The setup is simple, deploy a few Wireguard and FRR configs and have all the connectivity ready to go, set it and forget it.
  - And it finally works!

Attempt 3 – the good and the bad

- **The good:**
  - Well… It works!
  - No odd crashes, performs under load
  - With no HA, the maintenance and debugging is also simpler!
- **The bad:**
  - No HA: even though it was never a target, this is a little unfortunate

That’s it! Right…?

- **Almost! We have Openstack… now we need to solve the final few bits**
  - To make the VM accessible externally… We need public IPs. And well IPv4 is expensive!
  - It’s not very elastic, if we buy a /24 range we’d need to keep it for a while even if the lab environment isn’t being used.
  - As mentioned before… this is a common failure point, where unintentional breaches/abuse happens. So IP reputation, reports and all the other wonderful things become a bit of a concern!
  - Would be cool… but I don’t think we can pull that off safely.

So what do we do?

- **Put them all in a dark room and throw away the key!**
  Or in other words… a VPN!
  - This solves some problems:
    - We no longer need to own public IP ranges or worry about security there
    - The students are the only clearly authorized users, so any inbound connections regardless of their firewalls rule would only happen from within the VPN
    - It effectively functions as a protective bubble
  - But this also poses additional problems:
    - There's another moving piece, needing to walk through the setup, configuration on the client machine. As well as to make sure it keeps working!
    - And with direct external connectivity, something like LetsEncrypt needs "hacks" to work around these limitations

Solutions, solutions, solutions…

- **VPN installation on the client machines**
  - We decided this was acceptible and simply wrote decent documentation to address the client setup. While students still forget to turn it on sometimes, it’s a pretty minimal issue.
- **LetsEncrypt**
  - This was a little more tough, without being able to use HTTP-01 challenge, the only thing that was left is DNS-01
  - And let’s just say… wasn’t particularly thrilled to give a bunch of students access to edit DNS records for a domain.

DNS-01

- **This boils down to 2 main concerns:**
  - I need a way to safely delegate a (sub)domain to the students without compromising on security. Preferably that would work with any domain name provider.
  - And I’d want to integrate this back into Openstack. Asking the students to use a different interface for purely managing DNS is cumbersome, inconvenient and would lead to additional confusion.

But Openstack has Designate!

- **Thankfully for me, Openstack has a service called Designate, specifically for managing DNS zones!**
  - It gives me exactly what I need, a way to get DNS into Openstack for students to use!
  - But it became clear that most tooling doesn't have integration with designate. And those that do are often outdated/unsupported.
    - And no… I wasn't going to write one.
- **So I devised a bit of a "janky" DNS router…**

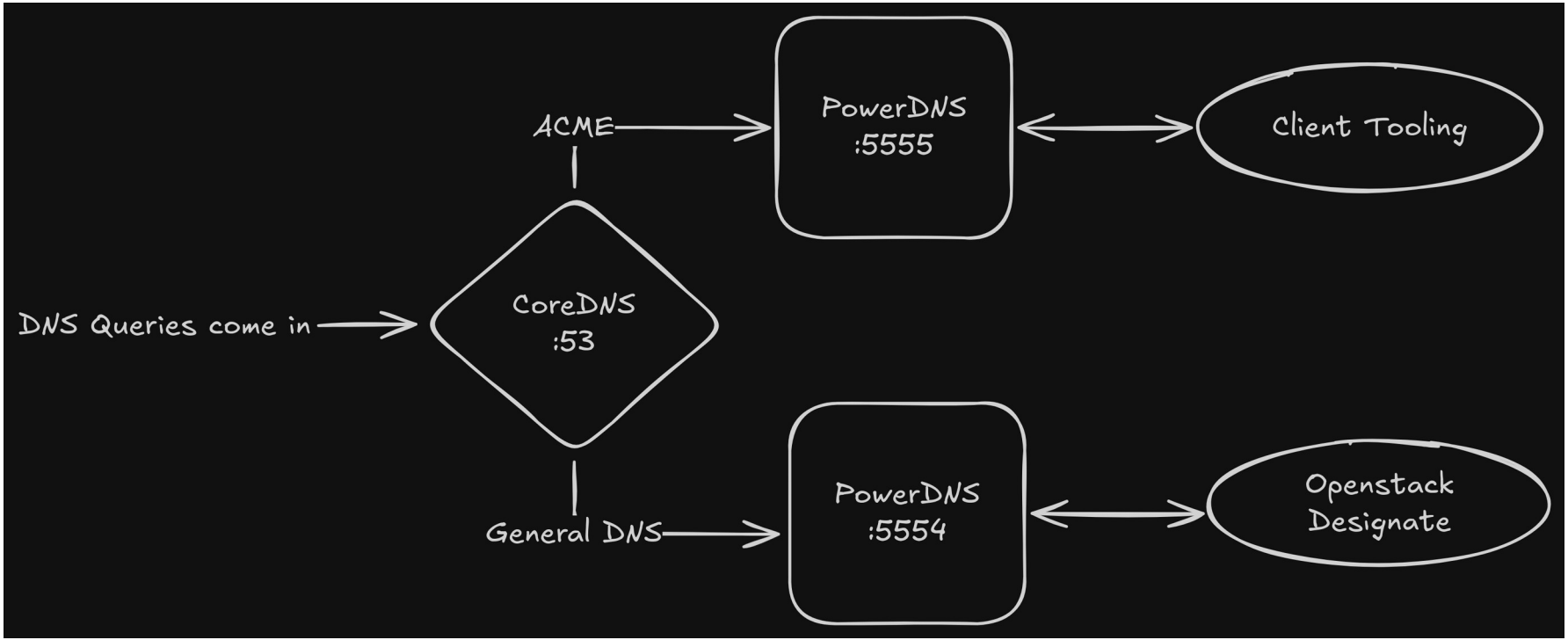
DNS-01… Embracing the “jank”!

- **For those who don’t know, DNS-01 depends on an _acme-challenge TXT record to exist and have a particular value for the domain you’re trying to get an SSL certificate for.**
- **So my idea was this:**
  - Openstack Designate works great for users. So let’s keep it for exactly what it is, UI and API.
  - But let’s use something that existing tooling supports for DNS-01 challenge:
    - I settled on running 2 PowerDNS resolvers: 1 for Designate to send changes to and 1 for LetsEncrypt verification
    - There’s a CoreDNS router standing in front of them, that forwards most queries to the PDNS resolver used for Designate, but any record that starts with _acme-challenge, gets sent to the second PDNS resolver.

DNS-01 – DNS go BRRRR!

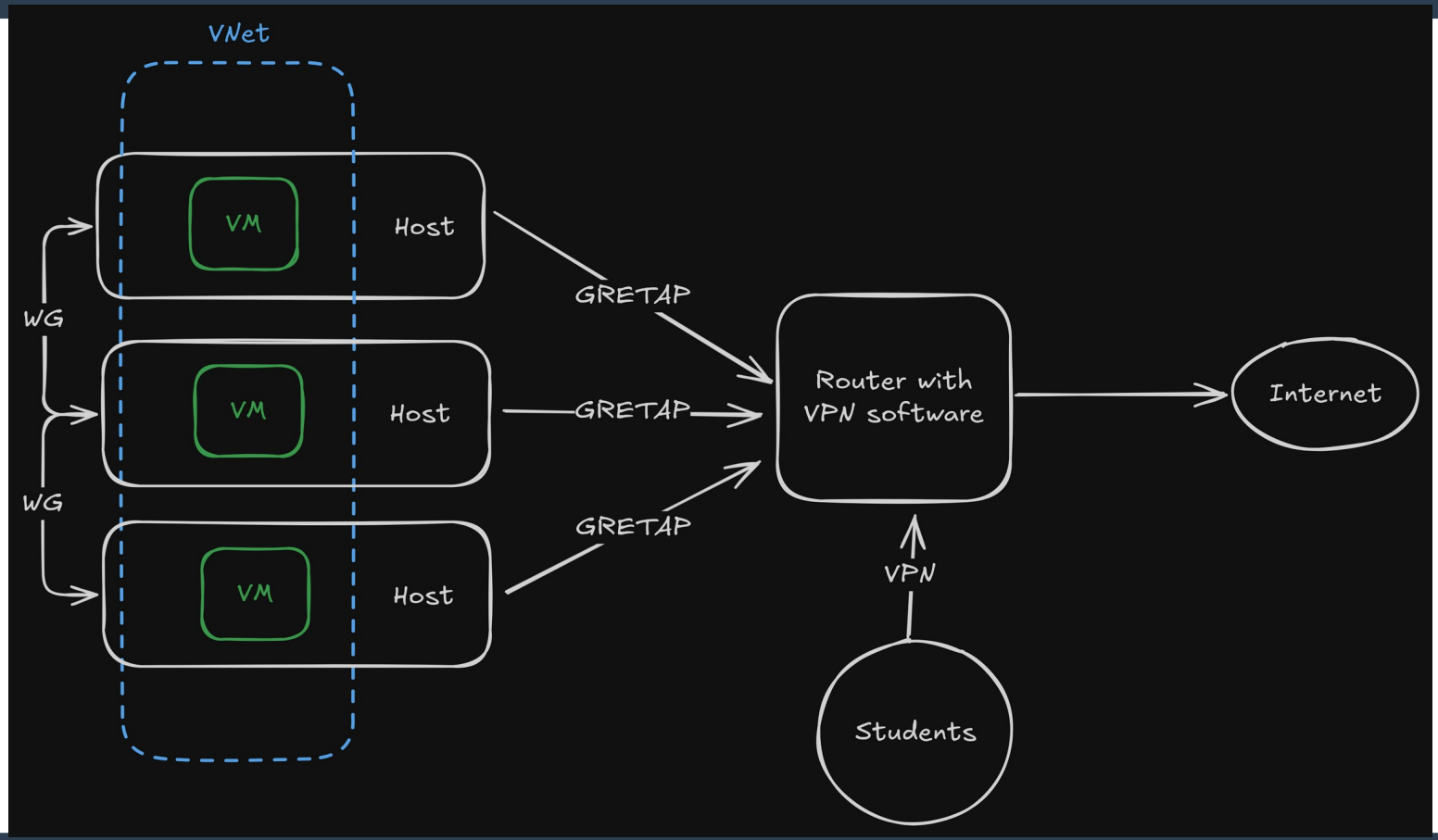
- **With this “routed” DNS setup, I get best of both worlds:**
  - Normal DNS records for IPs are done through Designate, with a nice frontend and API to do so. All in Openstack, nice and convenient.
  - While the client tooling used for can interact with the second PDNS resolver using the RFC2136 standard to automatically update DNS records for domain ownership verification
  - And basically all tools support RFC2136 in some manner

It looks a little like this



And finally we have a complete solution!

It’s not that bad!



Potential improvements for the future

- **HA could still be achieved with more component separation**
  - Keystone could just be put separately with the database and MQ, where it could be functional and performance. And have multiple “regions”
- **Networking could probably even be simplified further, Neutron has other networking plugins, but I haven’t tried them yet.**
- **Making the VPN gateway routed instead of using GRETAP for external connectivity, would allow for higher availability and better throughput.**
- **And finally… more documentation. Openstack isn’t ubiquitous knowledge and docs are often confusing or hard to follow. Writing down FAQs and common solutions to our wiki is how we’ll keep improving.**

So why “Jenga”?

- **First… Jenga is a trademark, so wood block game!**
- **And two, building cloud usually stands on top of a few great pillars (or blocks):**
  - Solid compute, preferably uniform
  - Solid networking, preferably higher than 1Gbit or at least not shared with the internet
  - Solid support – well… I’m just built different! Poorly.

Key takeaways

- **Opinionated solutions are great, until they are not… Maybe Microstack works great for some, but for me it was too limiting. *Too opinionated.***
- **Openstack is definitely not that scary to get going, but it will take significant time and effort. If you're going to attempt this… Be prepared, it'll take every ounce of knowledge to do well and support it long term.**
- **Invest in monitoring and automation here – it's a big machine, even with 3 hosts it can quickly get overwhelming tracking down all of the components and debugging.**

But we can do it!

- **But all that aside, this is absolutely doable, usable and achievable!**
  - I find that very cool, that we are at a time, where we can run our own private/public clouds. Especially in times where the whole global economy and trust is going through a bit of an overhaul!
- **So if you're interested in something like this for your school, business or personal project – feel free to reach out! As you can see… *It's EASY!***

Thank you!

If you have questions or just want to chat, please feel free to find me later.

Or send me an email at: **jonas@zetabitas.lt**