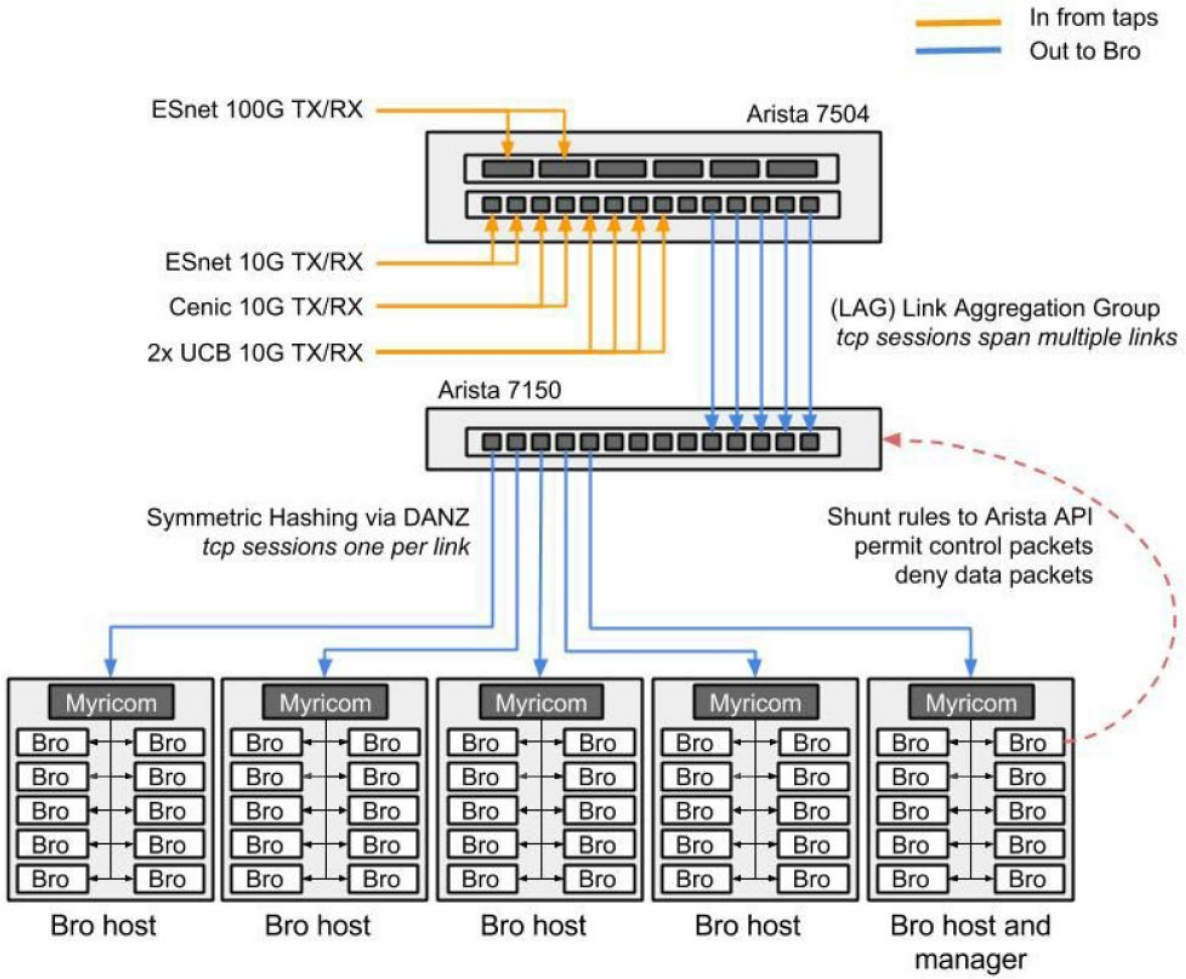
Managing Bro Deployments at Scale Using DevOps Technologies

Ed Sealing

Daniel Lohin

2015 Berkley Labs 100G Bro Cluster

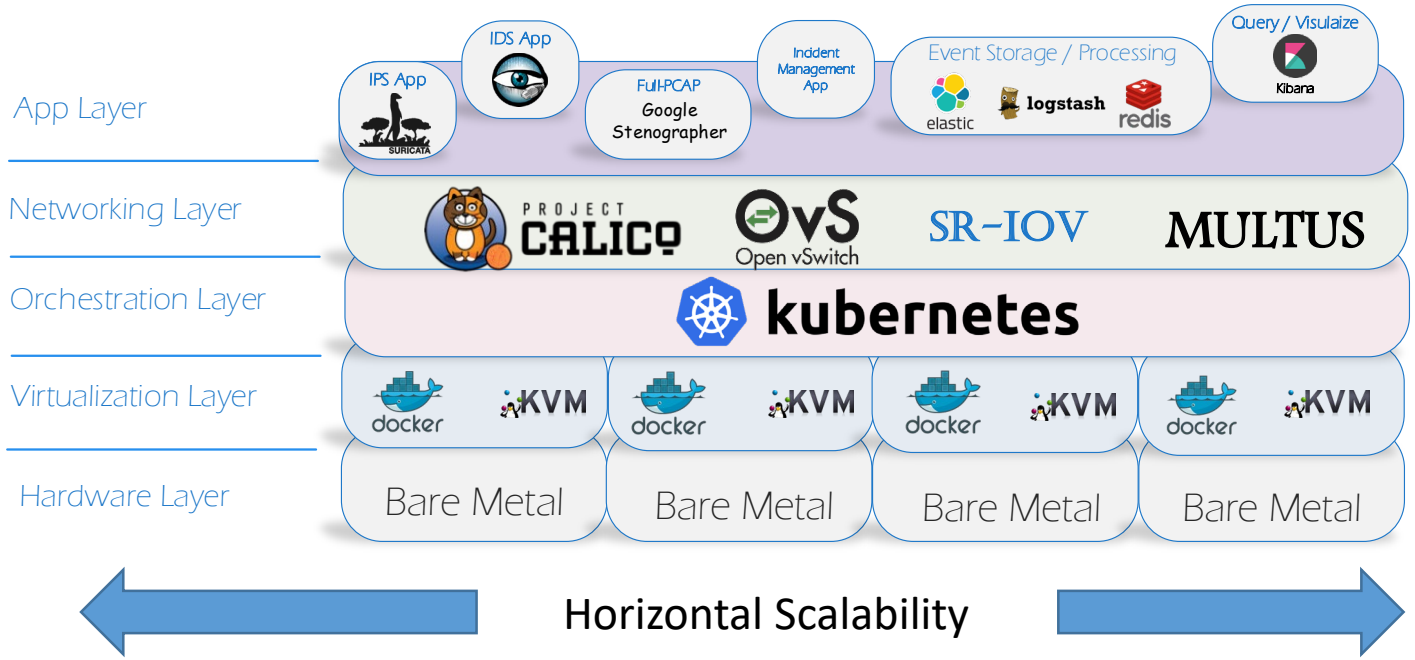


56 Node Bro Cluster
Paper: http://go.lbl.gov/100g

“Come on, this can’t be THAT hard…”

CONCEPT:

- Build Once, deploy anywhere
- Multi-Tenancy with resource segregation
- Shared Rules across mass cluster
- Shared Resources across different tools

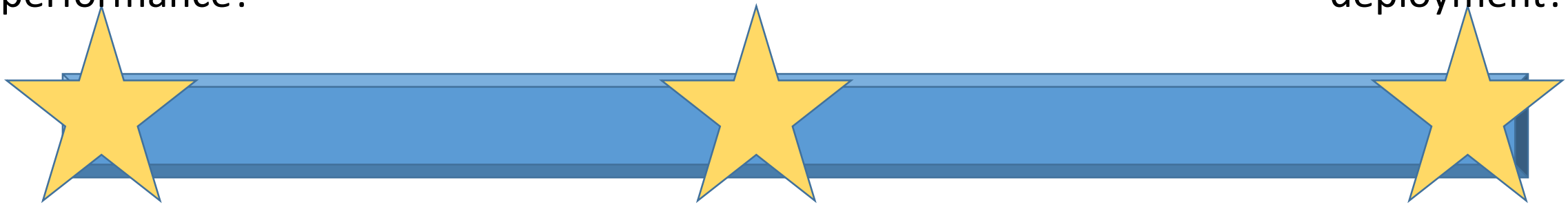


Our journey to enlightenment

Dec 2016-
Can we put Bro in a container and get decent performance?

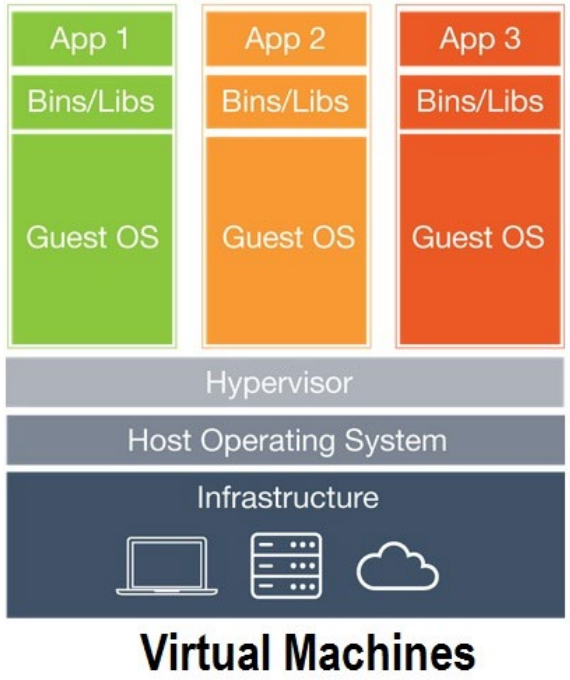
Summer 2017-
Can we automate deployment?

Summer 2018-
Can we automate a scalable deployment?

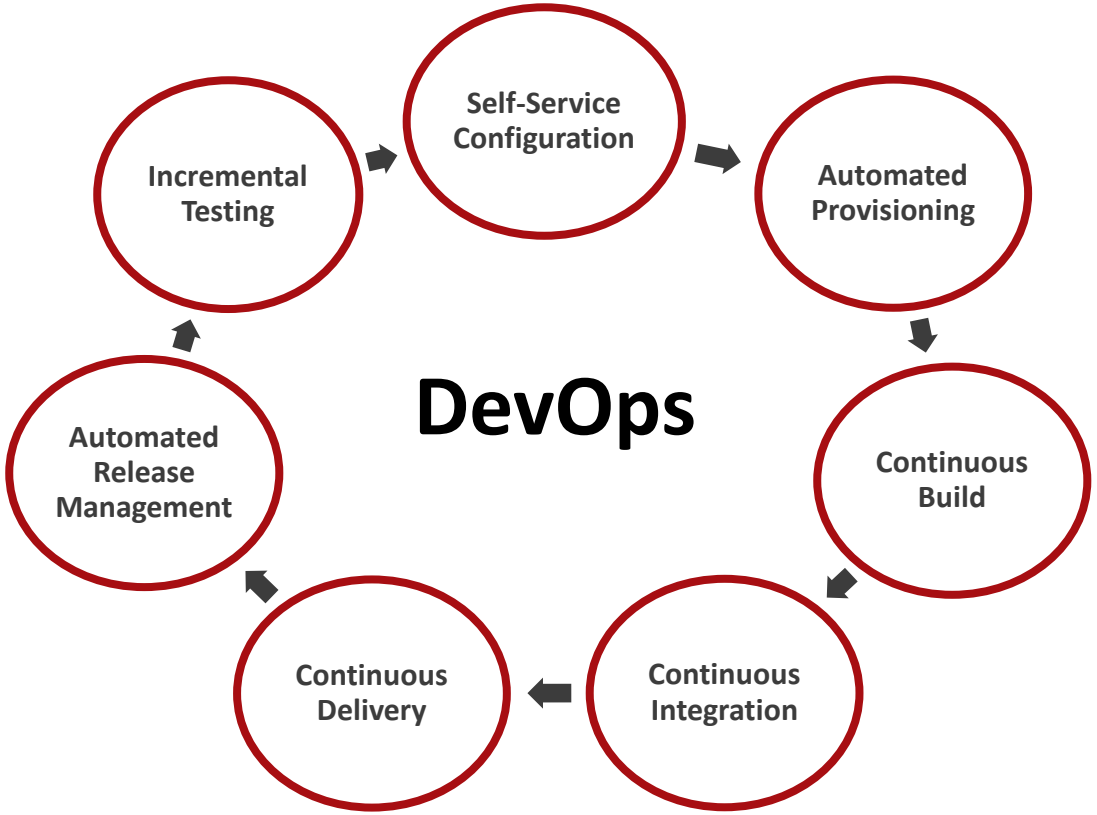


Why Containers and not VMs?

- Lightweight, stand-alone software that includes system tools, **system libraries** executable package.
- Packaged software for development, shipment as well as deployment
- **Containers share the machine's OS kernel**
- Containers are isolated using namespaces
  - PID
  - Networking
  - Mount Points
  - UID/GID
  - Limit processors and memory
  - And more!



DevOps Principals



Phase 1: Containerized Sensors perform?

- Chose two open-source network sensors (Bro & Suricata) and build DockerFiles for them
  - https://github.com/sealingtech/EDCOP-BRO
  - https://github.com/sealingtech/EDCOP-SURICATA
- What is the performance impact of running inside of a container?
  - https://www.bro.org/bro4pros2017/Sealing_Multi_Bro4Pros2017.pdf
- This image can be deployed again and again on different systems
- A lot of time was spent solving - How do we best get traffic to it?

Networking options we tried

Option	Description	Downside?
Host Networking	Give a container access to all networking on the physical host	Network isolation is gone. Container has complete control over all host networking.
MacVLAN/MacVTAP	Build to a physical interface and then connect a virtual interface to that bridge	Performance overhead
OpenVswitch	Build an openvswitch bridge and then create an interface with ovs-docker	Performance overhead and more complication
SR-IOV	Create a virtual NIC (called a Virtual Function) inside of the network card	Hardware dependent on this feature

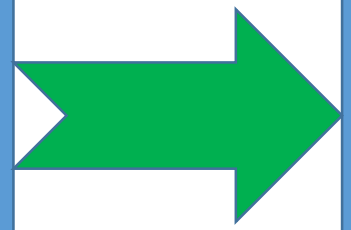
Lessons learned

- Hardware still matters… We still need to worry about IRQs, CPU pinning, NUMA nodes and all those other complicated things
- Containers are great for when you need to build an application on a single host, but what happens when you need to scale out to multiple hosts?
- We still didn't have integration with a larger architecture figured out (i.e. Bro feeding a Logging solution)… we needed more….
- Github or it didn't happen! https://github.com/sealingtech/bro-docker

Multi-stage containers

Build Container

Step 1. Install all build tools (GCC, Make, bro-pkg, etc)
Step 2. Build Bro
Step 3. Build all Bro Packages
Step 4. Start up the final image



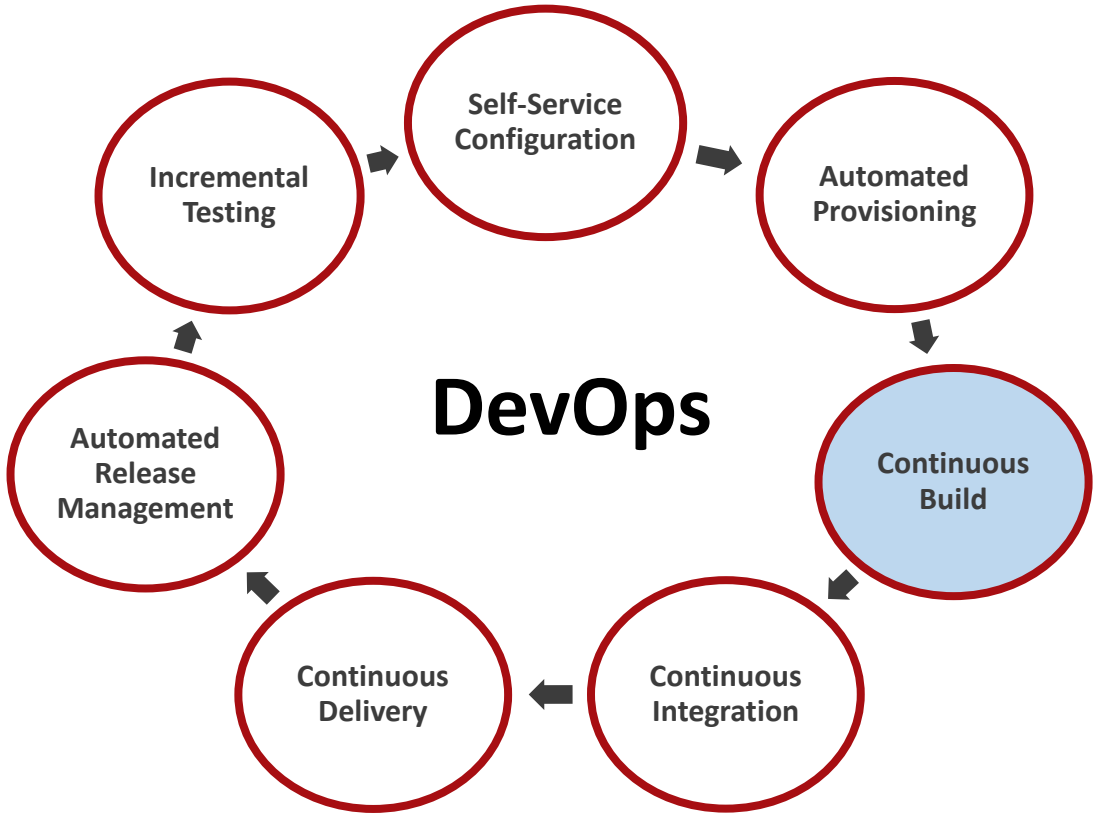
Final image

Step 1. Install packages only need to run Bro
Step 2. Copy final output of Bro from the build container
Step 3. Throw away the build container

- Bro can be built to get better performance
- Some Bro-packages require build tools
- Allows for containers to be smaller and prevents you from having to clean up!

https://github.com/dlohin/EDCOP-BRO/blob/master/container/Dockerfile

Phase 1 Progress

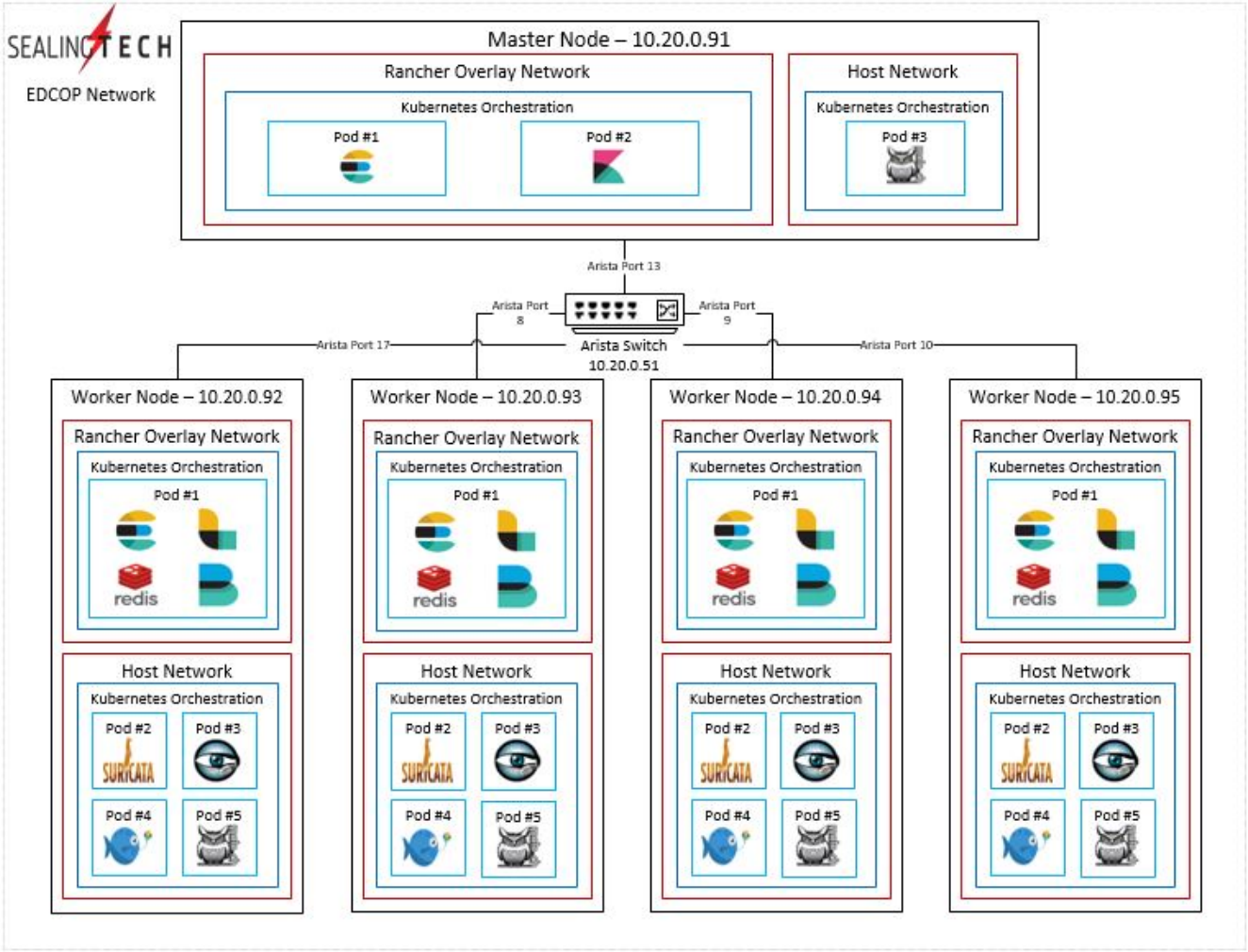


Phase 2: Automate an infrastructure around Bro

- Question: Now that we have a portable container, can we automatically deploy infrastructure around it?
- Answer: Yes! Our original proof-of-concept utilized Rancher to deploy Kubernetes and Bro.

Rancher Pros and Cons	
Pros:	**Cons:**
- Automatic infrastructure setup	- Limited customization
- Simple, easy to use	- Cluster management was a pain
- Variety of orchestrations supported	- Rely entirely on Rancher
- Could connect multiple nodes now!	- *Required use of host networking*

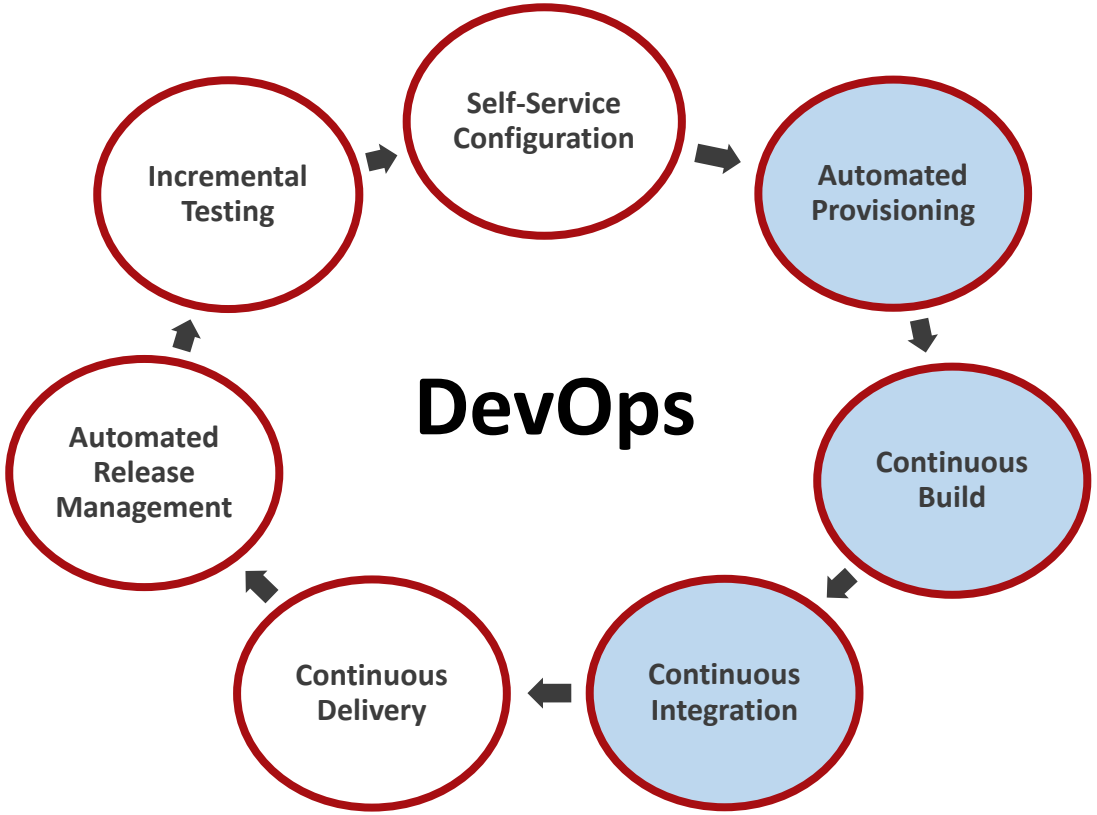
Proof of concept design



Lessons learned

- We were getting closer, but Rancher was designed to be flexible not customizable.
- The overlay network that Rancher used was a little interesting
- Rancher was used to deploy Kubernetes, I call this rancher-caption.. It is two container management solutions on top of one another
- NOTE: Rancher has changed a lot with 2.0, so I can't say if it has gotten better. They have moved to a more native Kubernetes platform

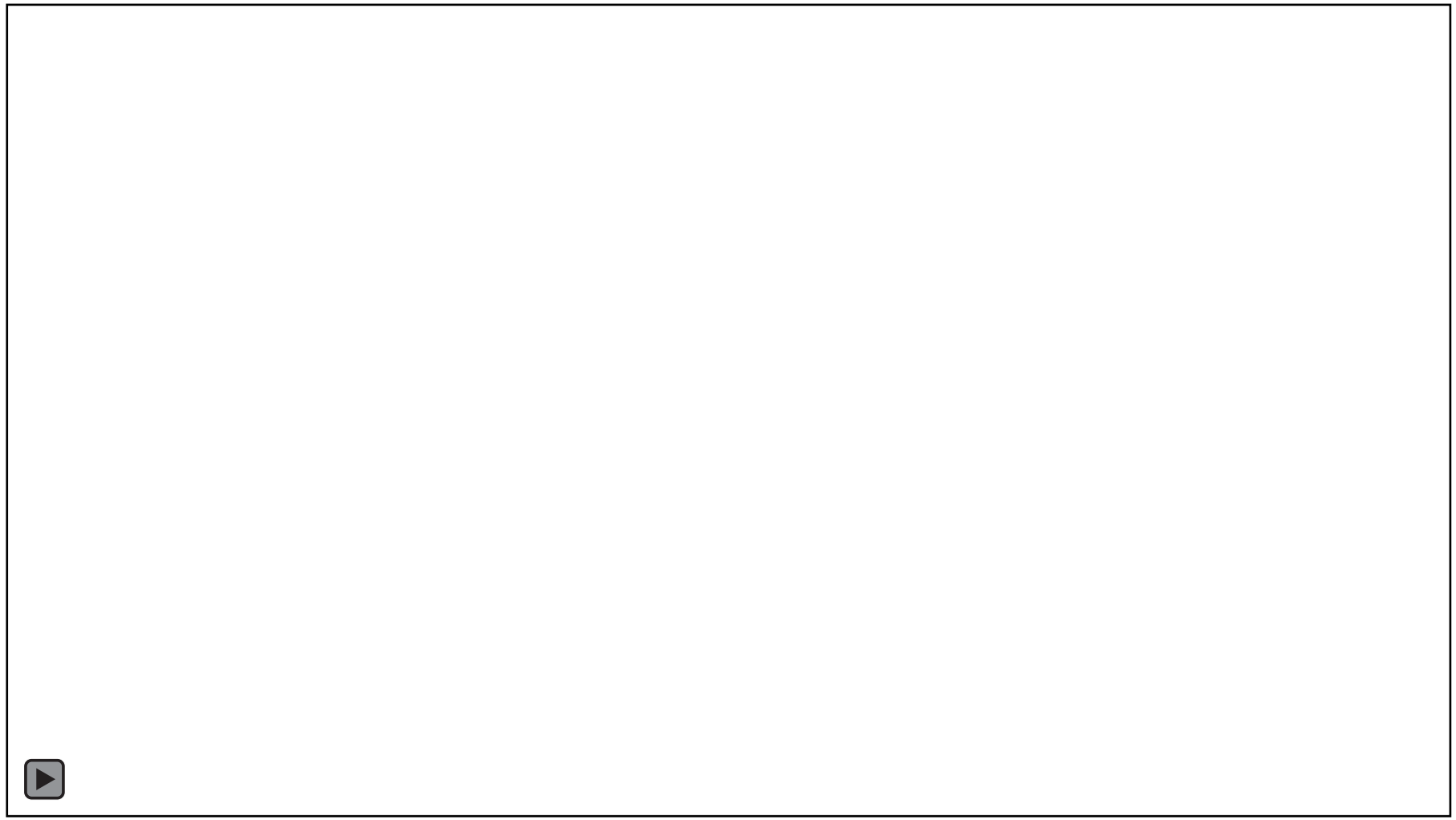
Phase 2 Progress



Phase 3: Build a scalable, customizable architecture

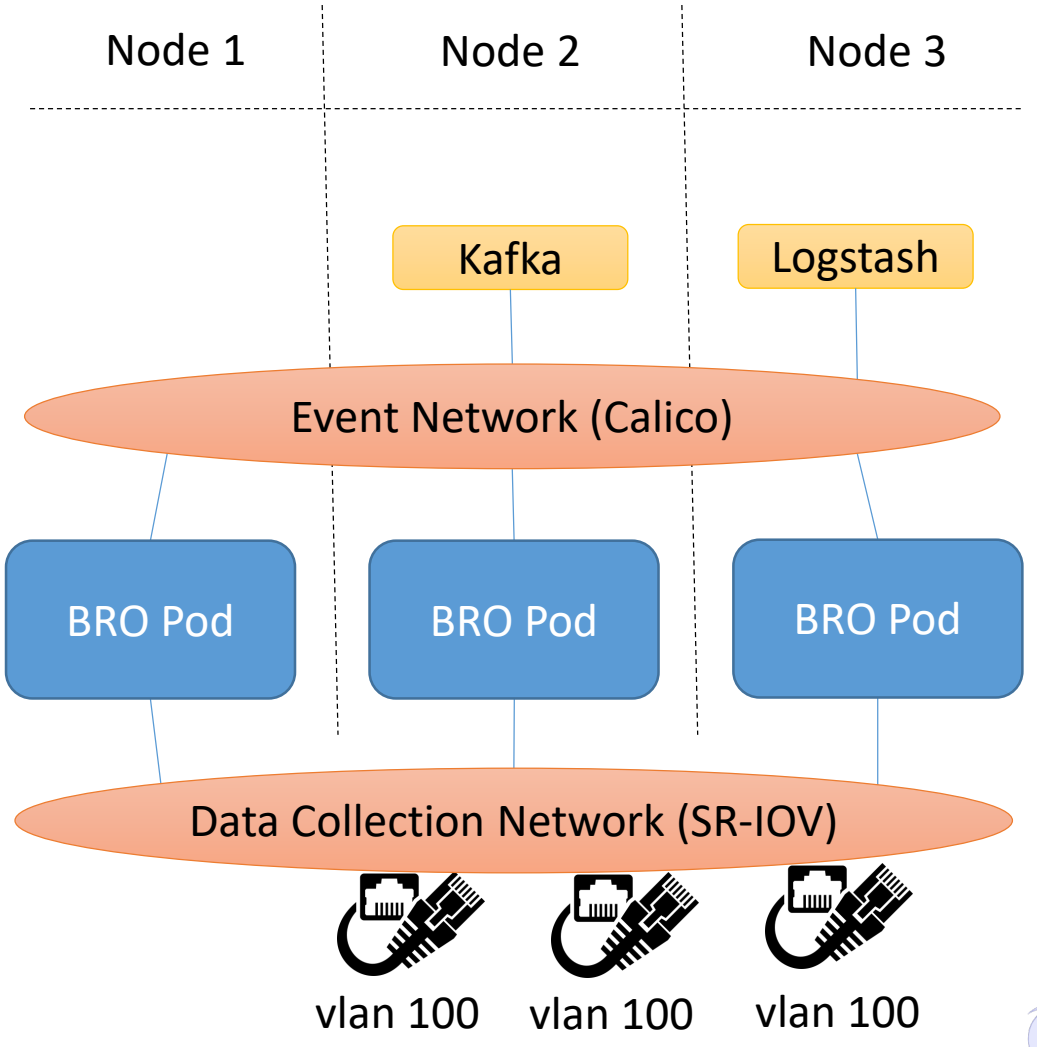
- We have containerized Bro and other sensors as well as the architecture around it
- Requirements
  - Need to be able to scale out, add more computers and applications can scale out accordingly
  - Traffic needs to be load balanced to allow sensors to scale
  - Services need to be customizable by end users
- Ability to utilize DevOps best practices

What it looks like…

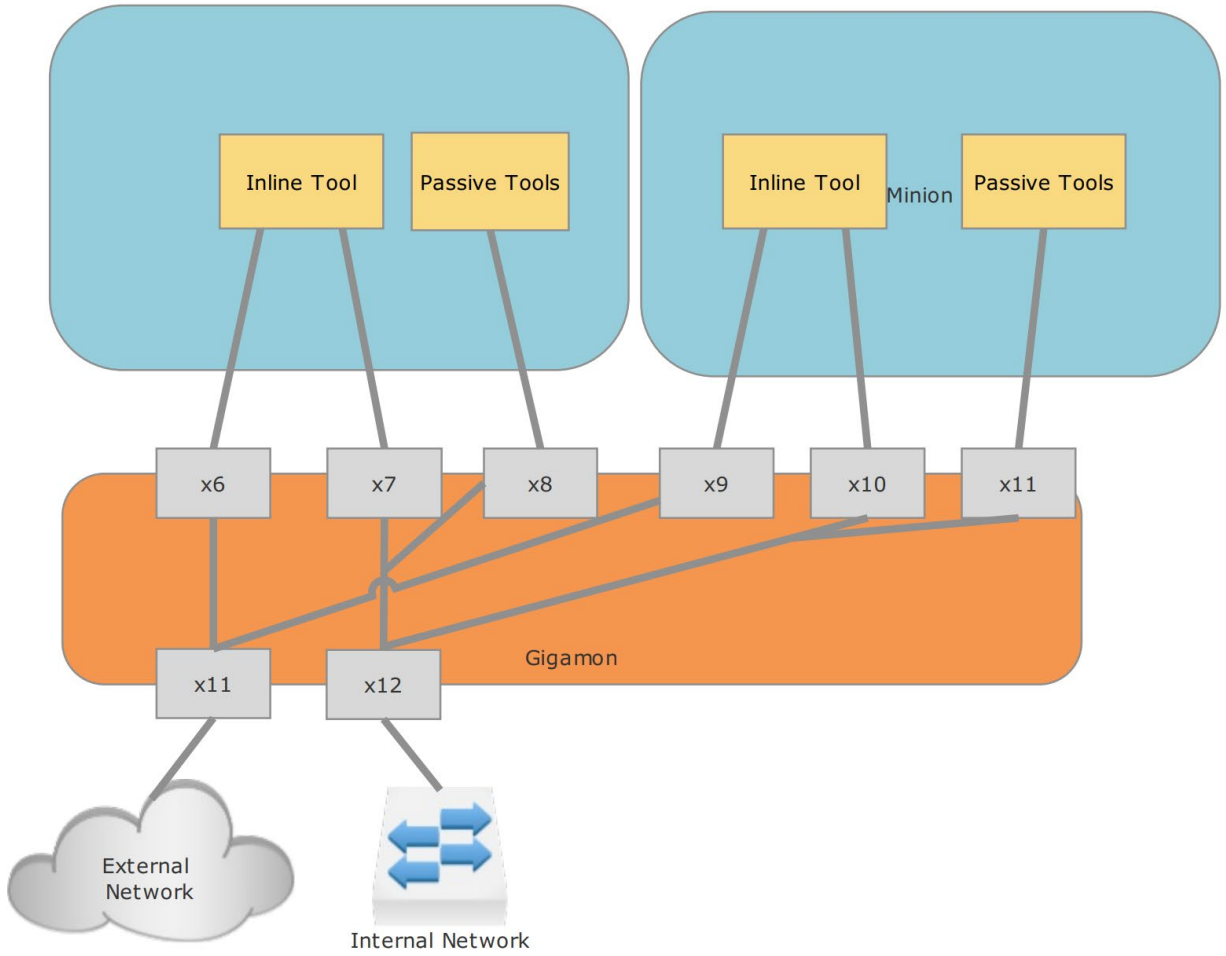


Problem 1: Multi-NIC containers

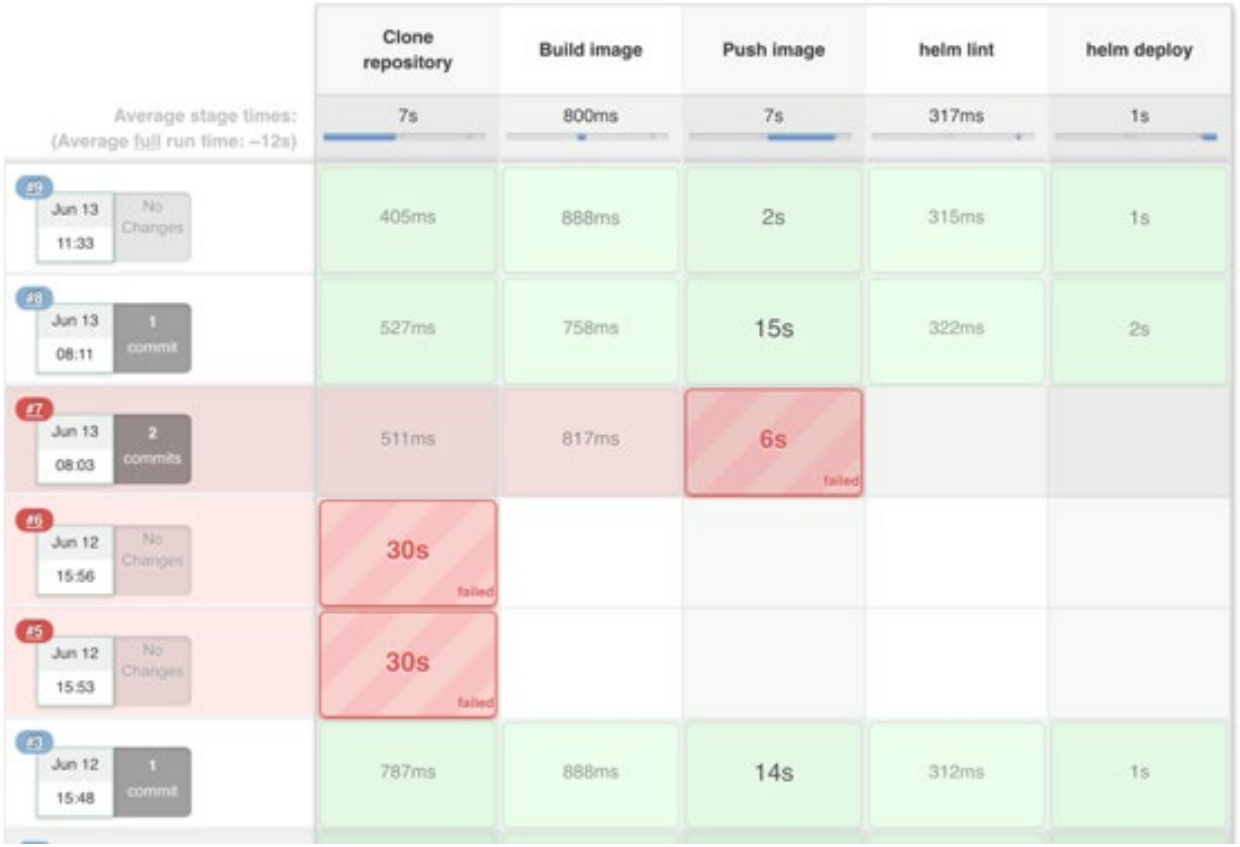
- By default, Kubernetes assumes you will have one network interface per pod
- Multus (an Intel project) allows multiple ETHs per pod on different networks



Traffic Acquisition

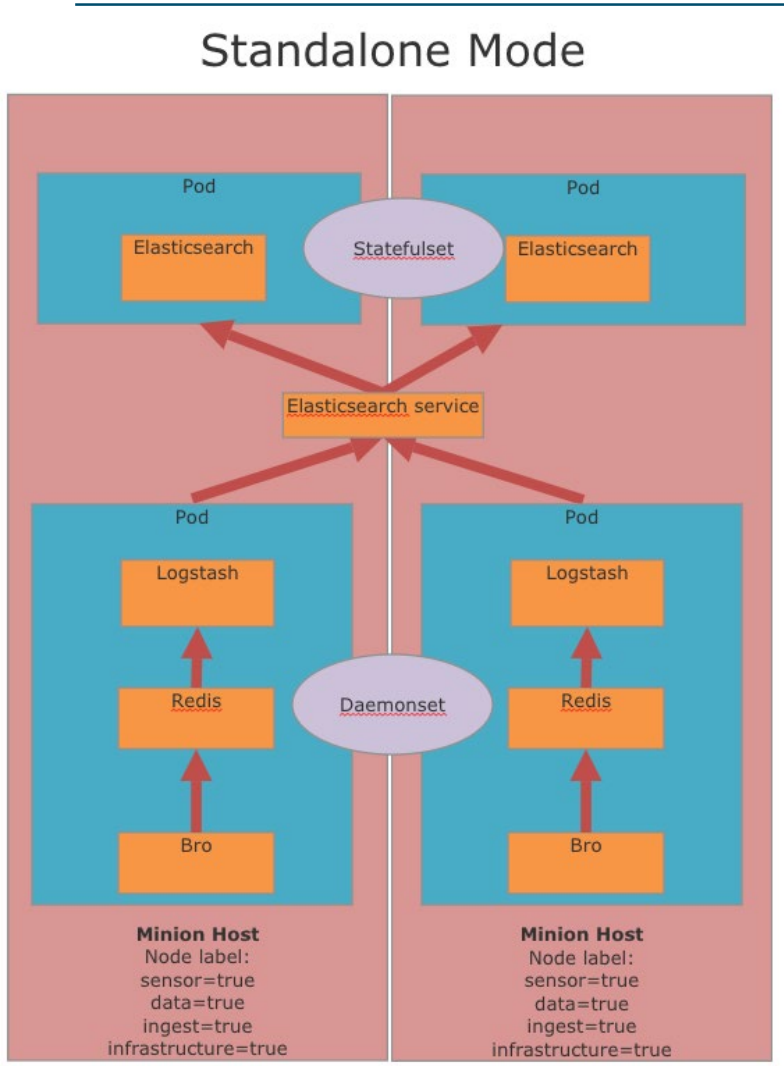


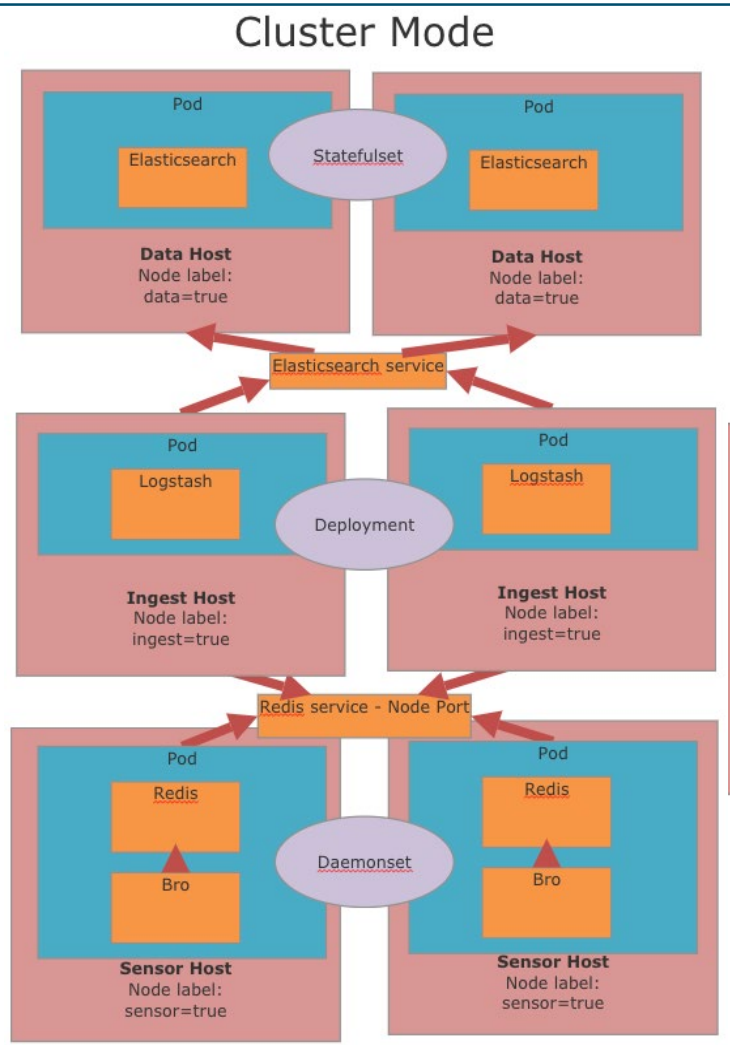
Jenkins Auto-Build of Bro using HELM

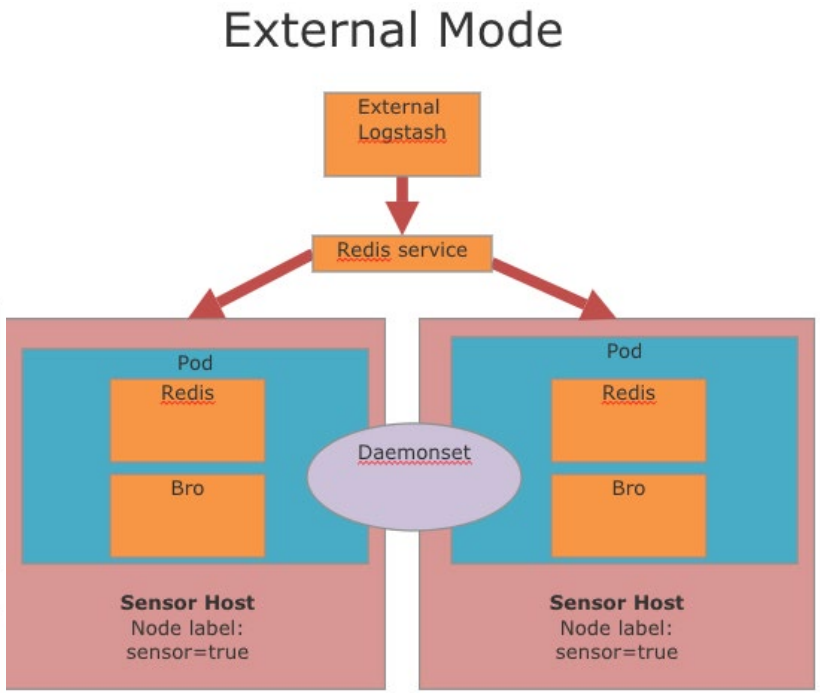


	Clone repository	Build image	Push image	helm lint	helm deploy
Average stage times: (Average full run time: ~12s)	7s	800ms	7s	317ms	1s
#9 Jun 13 11:33 No Changes	405ms	888ms	2s	315ms	1s
#8 Jun 13 08:11 1 commit	527ms	758ms	15s	322ms	2s
#7 Jun 13 08:03 2 commits	511ms	817ms	6s failed		
#6 Jun 12 15:56 No Changes	30s failed				
#5 Jun 12 15:53 No Changes	30s failed				
#3 Jun 12 15:48 1 commit	787ms	888ms	14s	312ms	1s

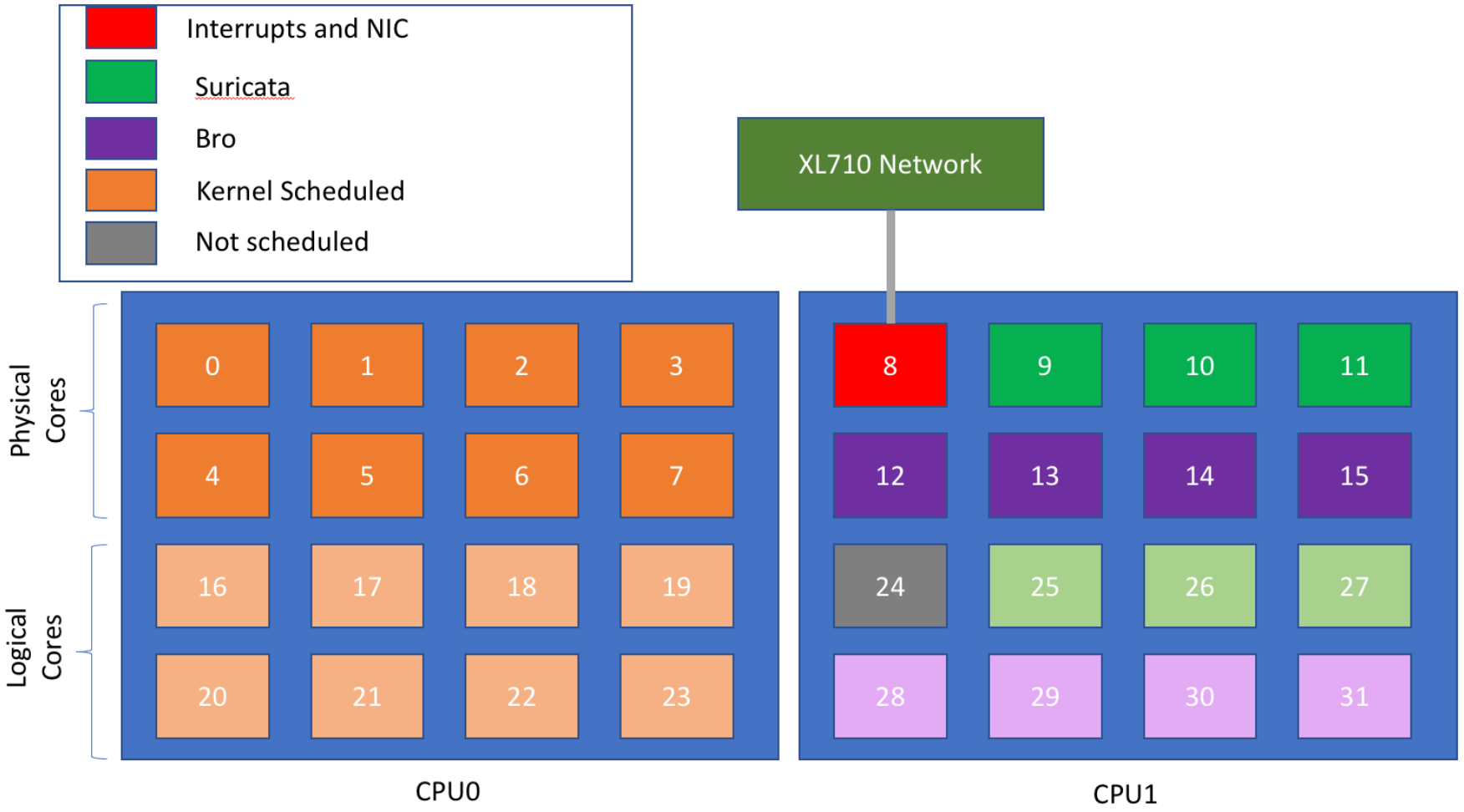
Deployment Options



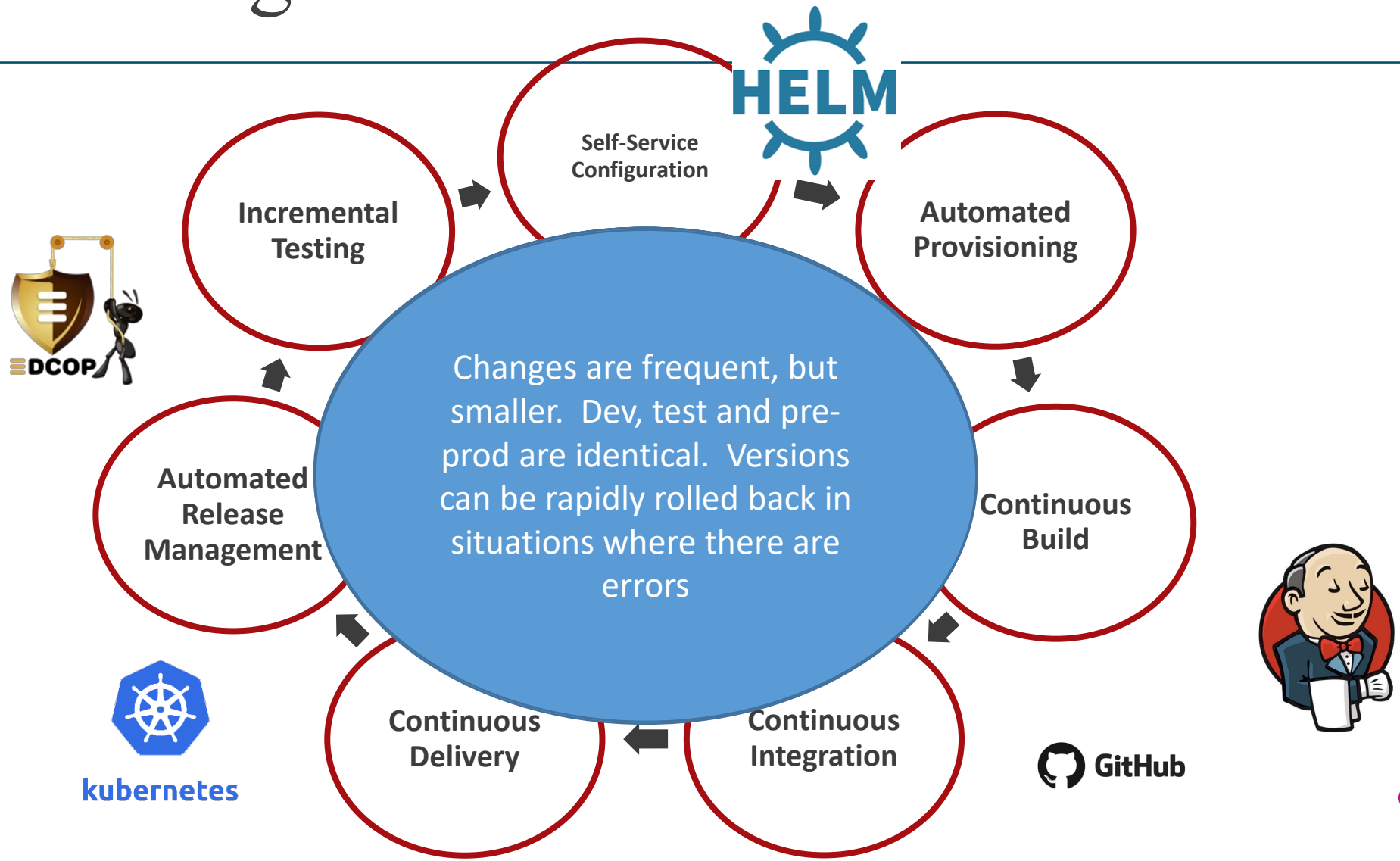




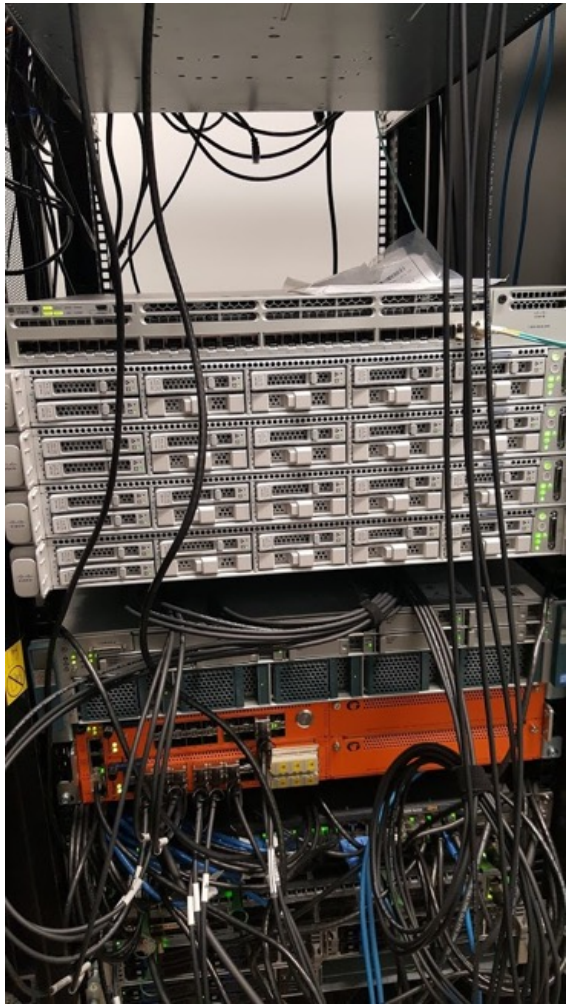
Compute resource management



Phase 3 Progress



Various iterations of testing



Lessons learned

- The Kubernetes community is moving incredibly quickly, every week there is some new cool way to do things… you can get caught chasing technology
- Designing an infrastructure around Kubernetes is a change in thinking. You learn to treat applications as temporary
- Stateless apps are a lot easier to handle then stateful apps
- Bro works great inside of Kubernetes you just need to plan

Show me the Github!!

- Website: https://edcop.io
- EDCOP Deployment Platform: https://github.com/sealingtech/EDCOP
- BRO: https://github.com/sealingtech/EDCOP-BRO
- All the other components are in seperare repos, just look for EDCOP-<tool name> here: https://github.com/sealingtech/
- Contact us:
  - ed.sealing@sealingtech.com
  - daniel.lohin@sealingtech.com