Supporting Docker in Emulab-Based Network Testbeds

David Johnson, Elijah Grubb, **Eric Eide**
University of Utah

l'étranger @cmeik · May 11

If anyone has some extra Amazon or Google credits that they are willing to donate — I have a master's student who is going to evaluate scaling Erlang/Elixir applications with Partisan to 4K nodes. We could help funding this, if you have credits lying around!

3 10 5

l'étranger @cmeik · May 11

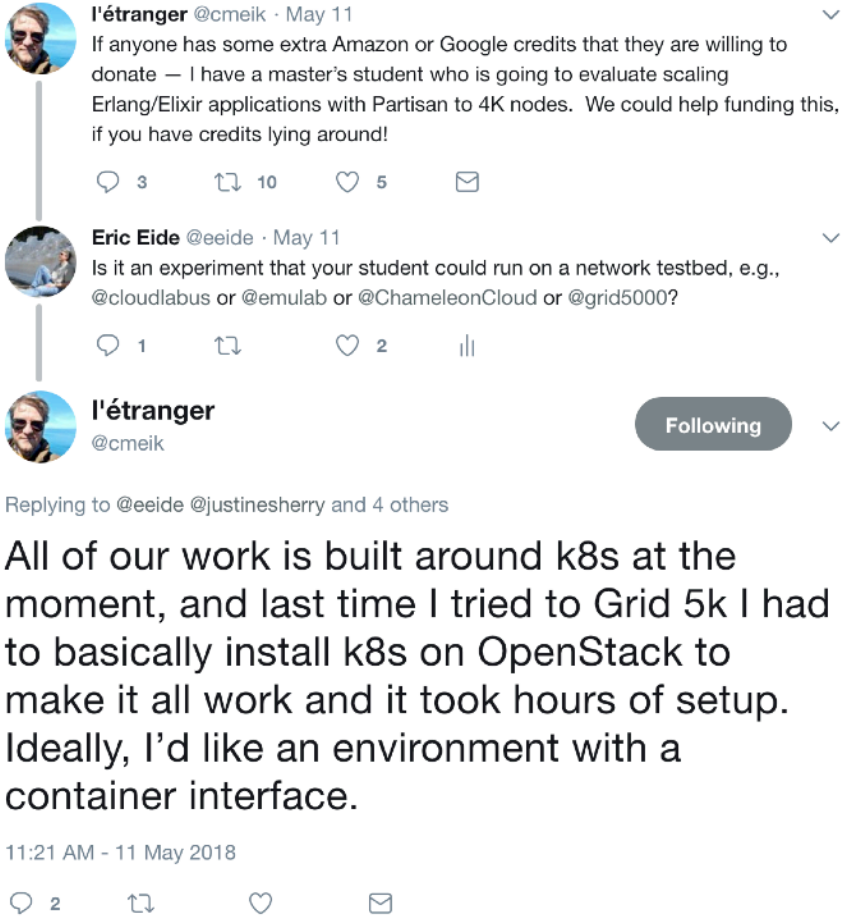
If anyone has some extra Amazon or Google credits that they are willing to donate — I have a master's student who is going to evaluate scaling Erlang/Elixir applications with Partisan to 4K nodes. We could help funding this, if you have credits lying around!

3 10 5

Eric Eide @eeide · May 11

Is it an experiment that your student could run on a network testbed, e.g., @cloudlabus or @emulab or @ChameleonCloud or @grid5000?

1 2


l'étranger @cmeik · May 11
If anyone has some extra Amazon or Google credits that they are willing to donate — I have a master's student who is going to evaluate scaling Erlang/Elixir applications with Partisan to 4K nodes. We could help funding this, if you have credits lying around!
3
10
5
Eric Eide @eeide · May 11
Is it an experiment that your student could run on a network testbed, e.g., @cloudlabus or @emulab or @ChameleonCloud or @grid5000?
1
2
l'étranger
@cmeik
Following
Replying to @eeide @justinesherry and 4 others
All of our work is built around k8s at the moment, and last time I tried to Grid 5k I had to basically install k8s on OpenStack to make it all work and it took hours of setup. Ideally, I'd like an environment with a container interface.
11:21 AM - 11 May 2018
2


l'étranger @cmeik · May 11
If anyone has some extra Amazon or Google credits that they are willing to donate — I have a master's student who is going to evaluate scaling Erlang/Elixir applications with Partisan to 4K nodes. We could help funding this, if you have credits lying around!
3
10
5
Eric Eide @eeide · May 11
Is it an experiment that your student could run on a network testbed, e.g., @cloudlabus or @emulab or @ChameleonCloud or @grid5000?
1
2
l'étranger
@cmeik
Following
Replying to @eeide @justinesherry and 4 others
All of our work is built around k8s at the moment, and last time I tried to Grid 5k I had to basically install k8s on OpenStack to make it all work and it took hours of setup. Ideally, I'd like an environment with a container interface.
11:21 AM - 11 May 2018
2


l'étranger @cmeik · May 11
Also, I burned almost my entire lease time on just setup in the environment before even getting to the experiment.
1
l'étranger
@cmeik
Following
Replying to @cmeik @eeide and 5 others
Positives and negatives: building on containers got our software industry adoption, but now it's more difficult and more time consuming to run on most of the academic environments like CloudLab and Grid5k.
11:22 AM - 11 May 2018

- over the course of a study…
  - prototype on laptop
  - network testbed
  - commercial cloud
- need to move experimental artifacts around

- over the course of a study…
  - prototype on laptop 

  - network testbed
  - commercial cloud
- need to move experimental artifacts around

- over the course of a study…
  - prototype on laptop 

  - network testbed
  - commercial cloud 

- need to move experimental artifacts around

- over the course of a study…
  - prototype on laptop
  - network testbed
  - commercial cloud

- need to move experimental artifacts around

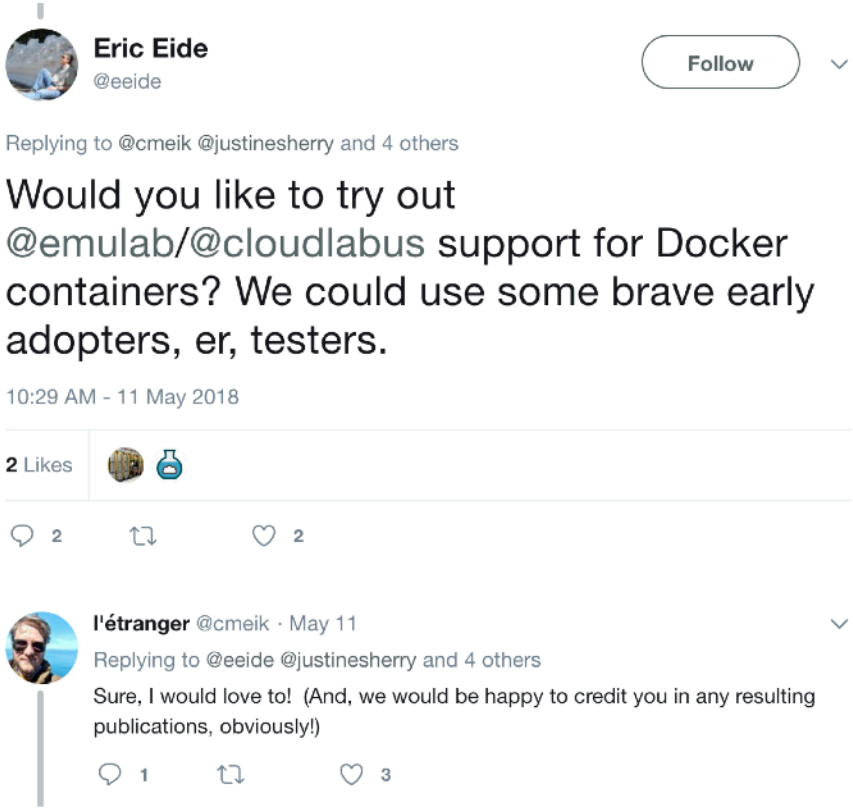
- over the course of a study…
  - prototype on laptop 

  - network testbed 

  - commercial cloud 

- need to move experimental artifacts around

- over the course of a study…
  - prototype on laptop
  - network testbed
  - commercial cloud
- need to move experimental artifacts around



This talk

- extended Emulab so users can create experiments in which some or all nodes are Docker containers

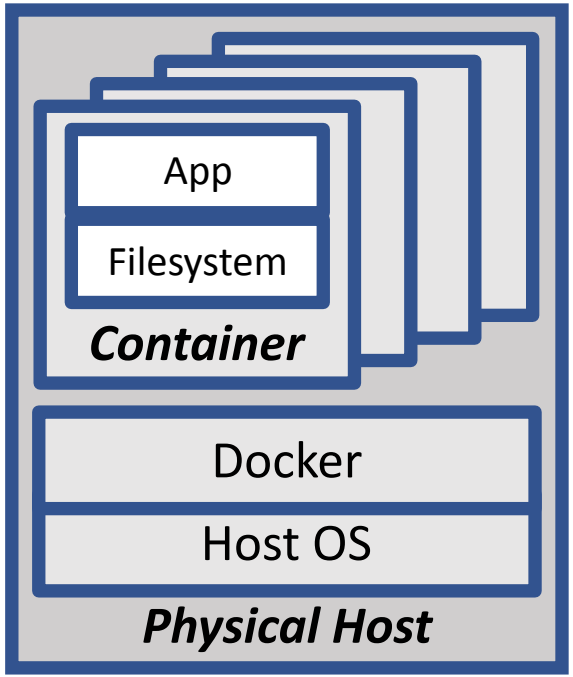
- challenges
  - preserving users' "testbed experience"
  - meshing with Emulab's infrastructure

- results
  - **just works:** 52/60 top Docker Hub images **automatically adapted**
  - supports large (5K-node) experiments

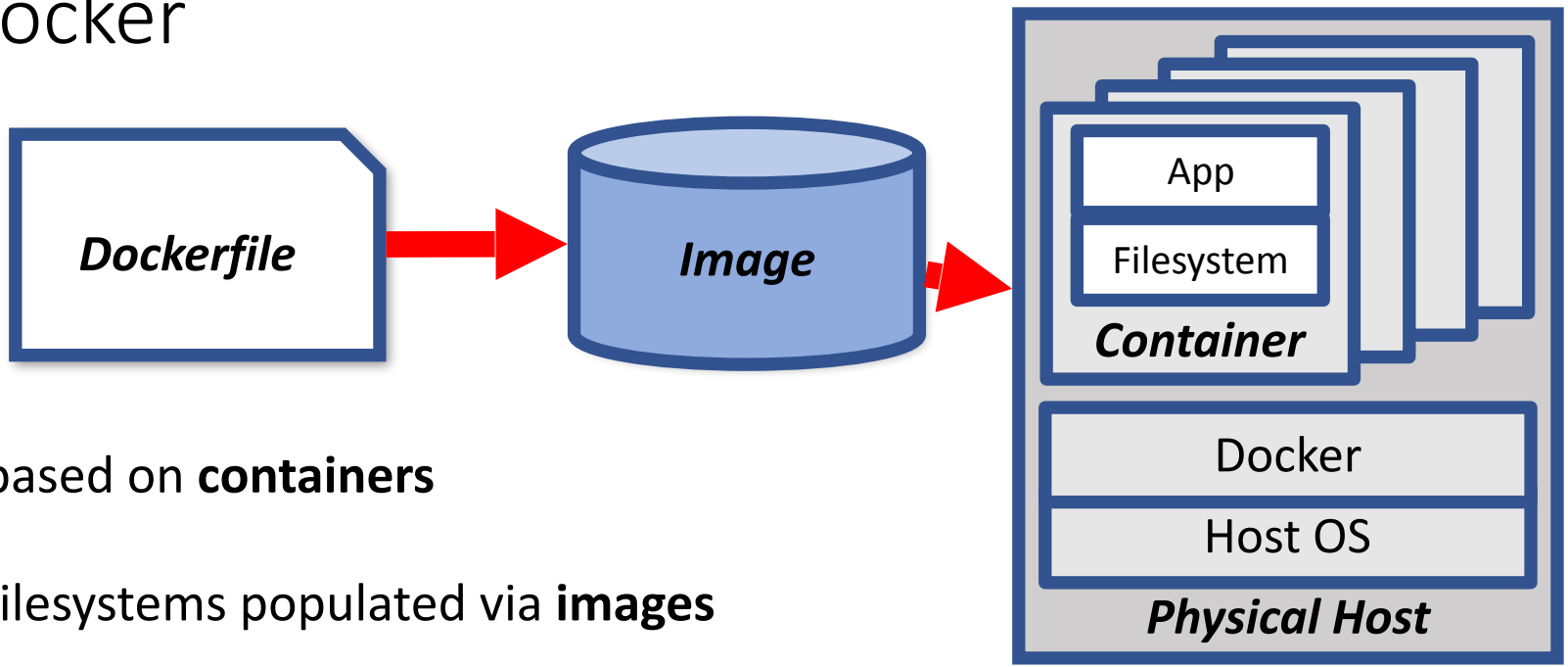
Docker

Docker

- based on **containers**
- filesystems populated via **images**



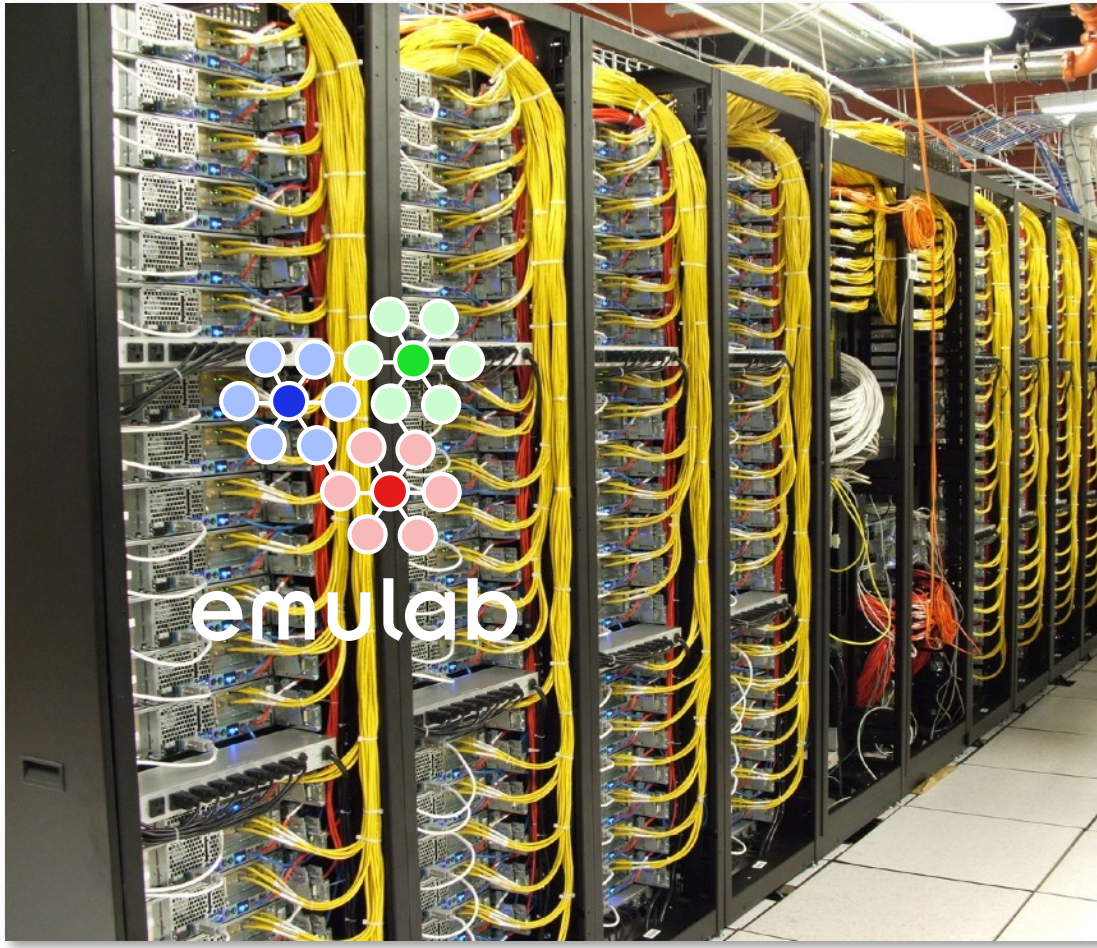
Docker



- based on **containers**
- filesystems populated via **images**
- images created via **`Dockerfiles`**

Emulab

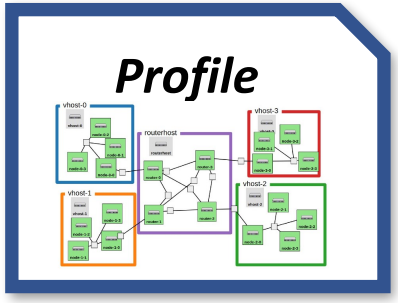
- testbed management software
- **allocates** physical and virtual resources to users
- **configures** resources
- **isolates** users from each other



Emulab

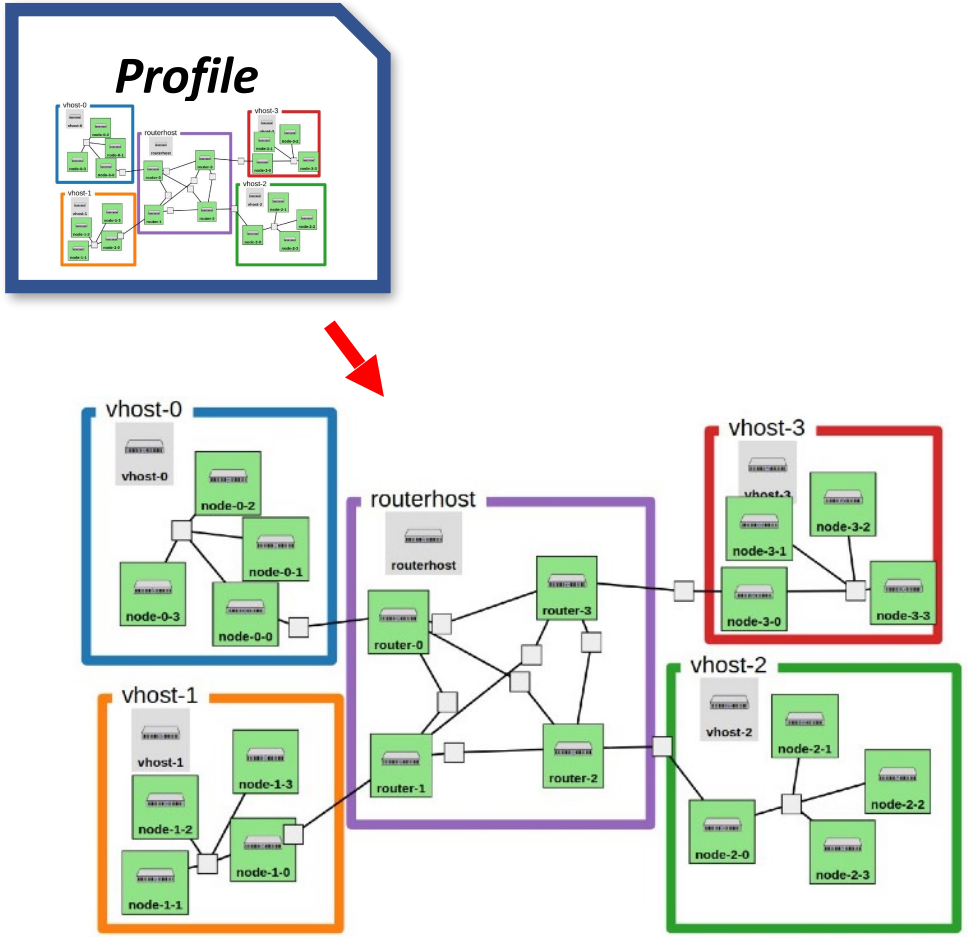
Emulab

- organized around **profiles**
- profiles are instantiated to make **experiments**



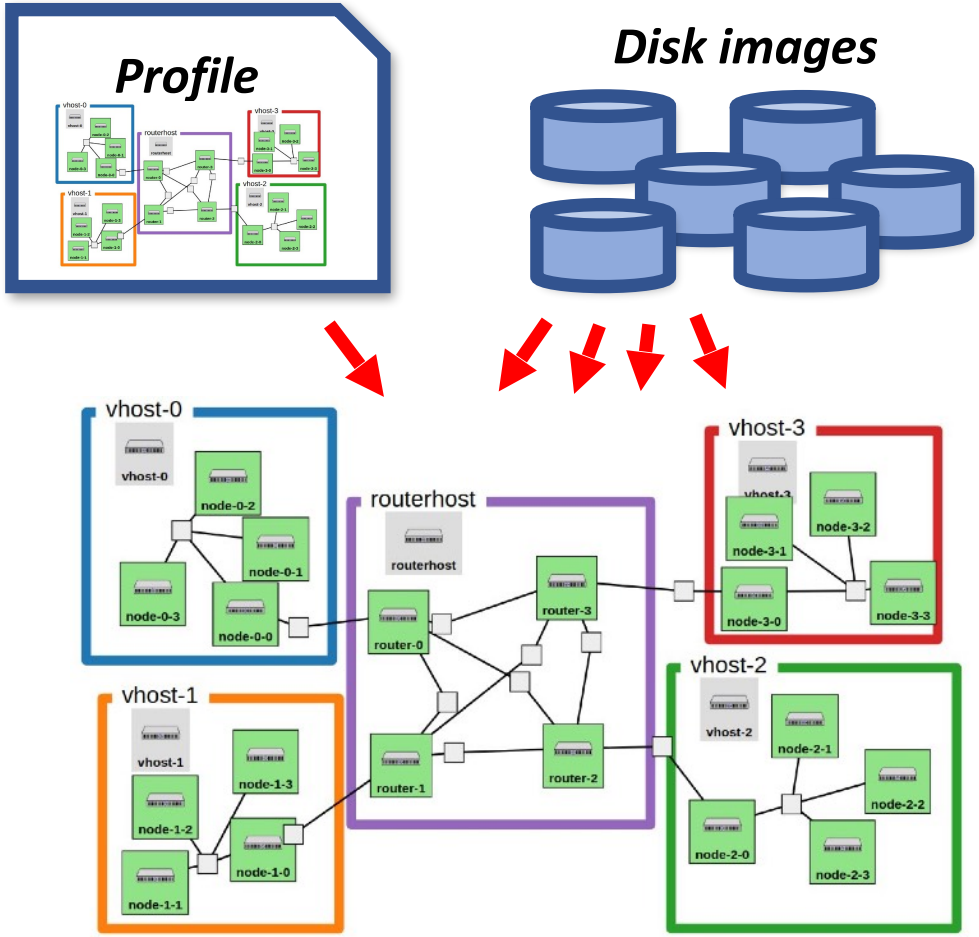
Emulab

- organized around **profiles**
- profiles are instantiated to make **experiments**
- nodes' disks populated via **disk images**



Emulab

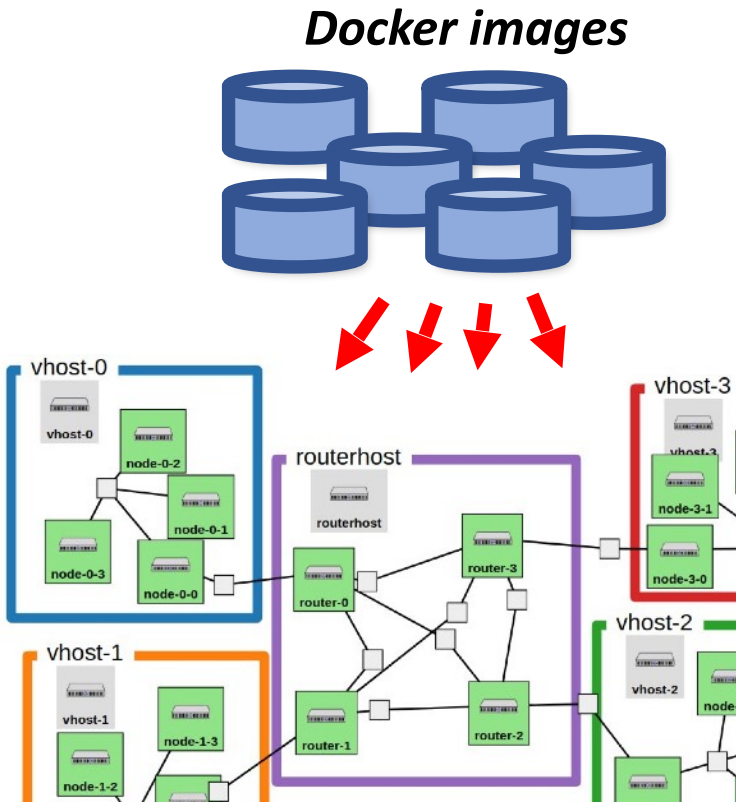
- organized around **profiles**
- profiles are instantiated to make **experiments**
- nodes' disks populated via **disk images**
- in-experiment **services**



Goal: Emulab + Docker should “just work”

Goal: Emulab + Docker should “just work”

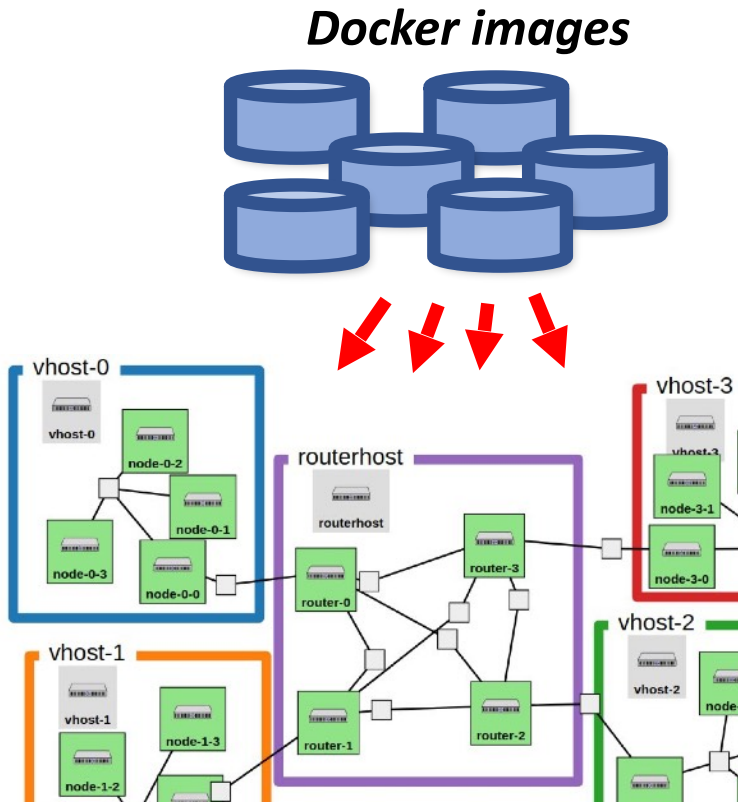
- containers in Emulab are just another kind of virtual node



Goal: Emulab + Docker should “just work”

- containers in Emulab are just another kind of virtual node
- Emulab user can choose any Docker image

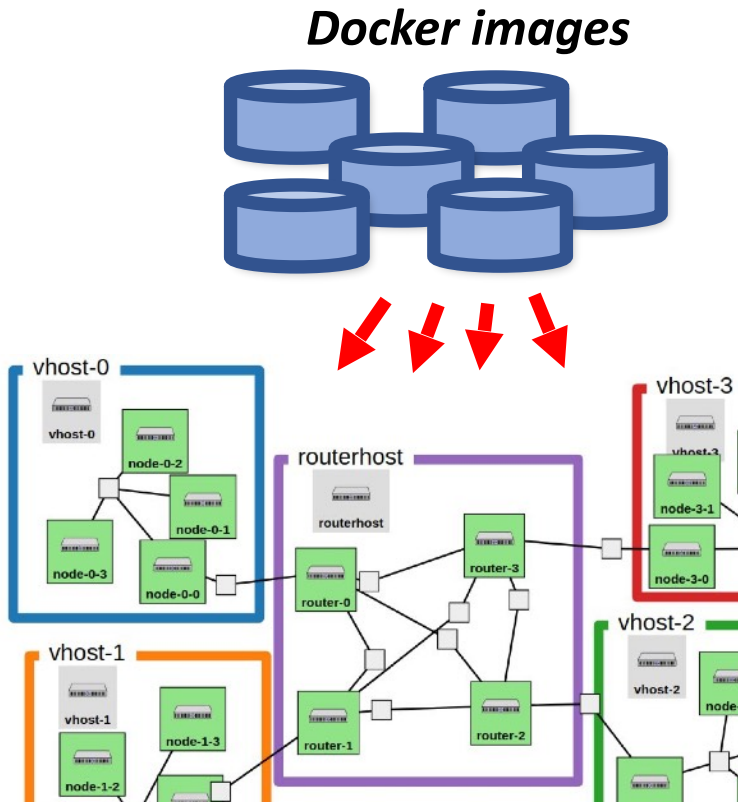
- preserve Emulab’s experimenter services
  - e.g., SSH, local/remote storage access, …



Goal: Emulab + Docker should “just work”

- containers in Emulab are just another kind of virtual node
- Emulab user can choose any Docker image

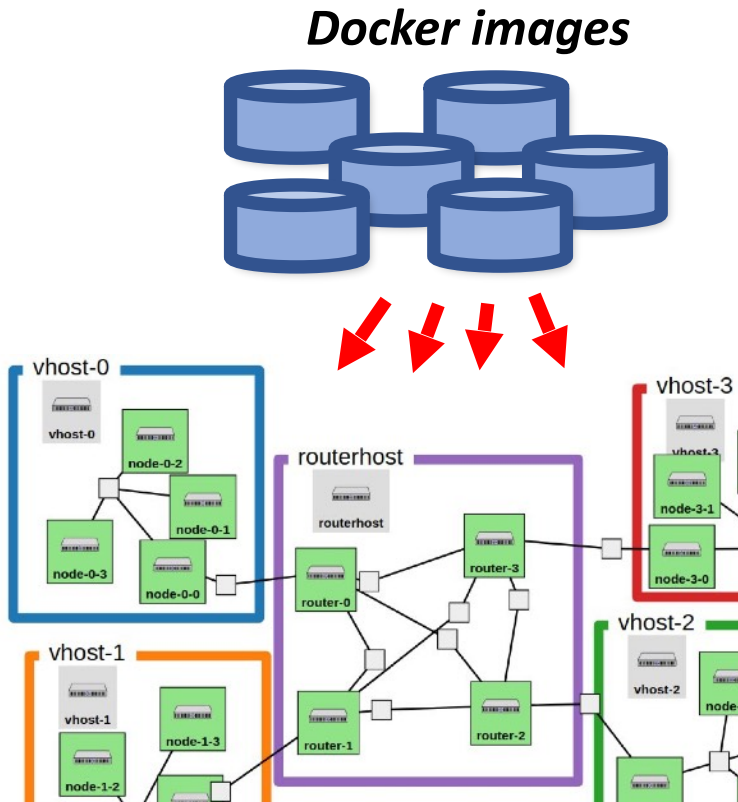
- preserve Emulab’s experimenter services
  - e.g., SSH, local/remote storage access, …
- preserve Emulab’s network services
  - e.g., control network, traffic shaping, …

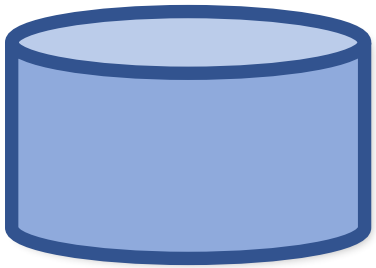


Goal: Emulab + Docker should “just work”

- containers in Emulab are just another kind of virtual node
- Emulab user can choose any Docker image

- preserve Emulab’s experimenter services
  - e.g., SSH, local/remote storage access, …
- preserve Emulab’s network services
  - e.g., control network, traffic shaping, …
- preserve Docker user experience
  - e.g., “`docker commit`”





`httpd:latest`

Preserving Emulab's experimenter services

- shell access to nodes
- remote and local storage
- network configuration
  - addressing, routing, shaping
- startup programs

Preserving Emulab's experimenter services

- shell access to nodes
- remote and local storage
- network configuration
  - addressing, routing, shaping
- startup programs

- typical Docker images are minimal **appliances**

- **run the application only**
- not prepared to host other services

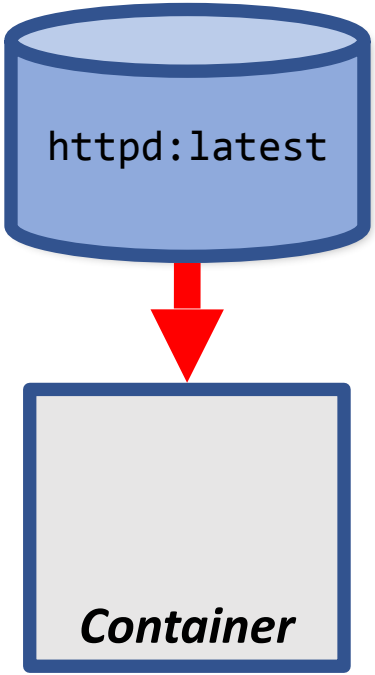
augmentation

generate a new `Dockerfile`,
starting from the user's chosen image,
and adding testbed software

Augment the startup

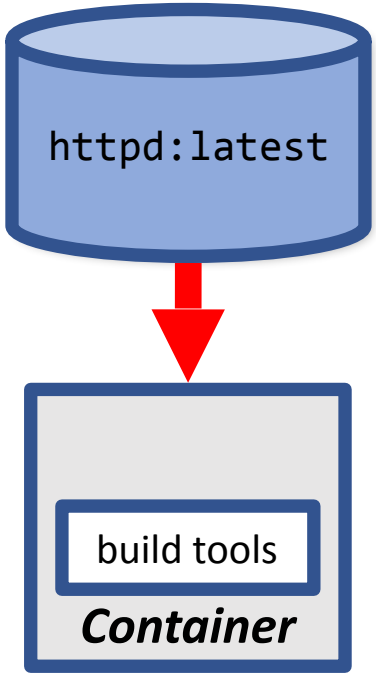
Augment the startup

- make temporary container



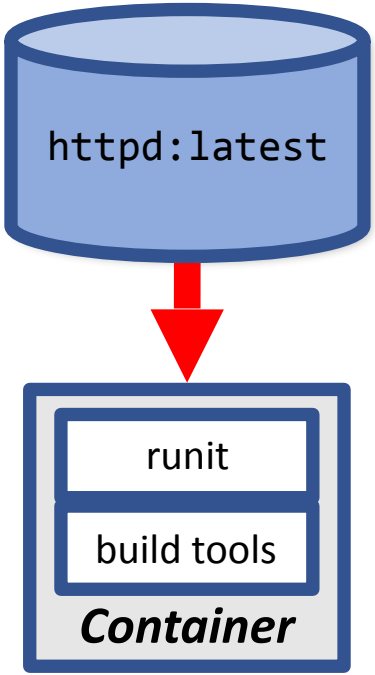
Augment the startup

- make temporary container
- add build toolchain



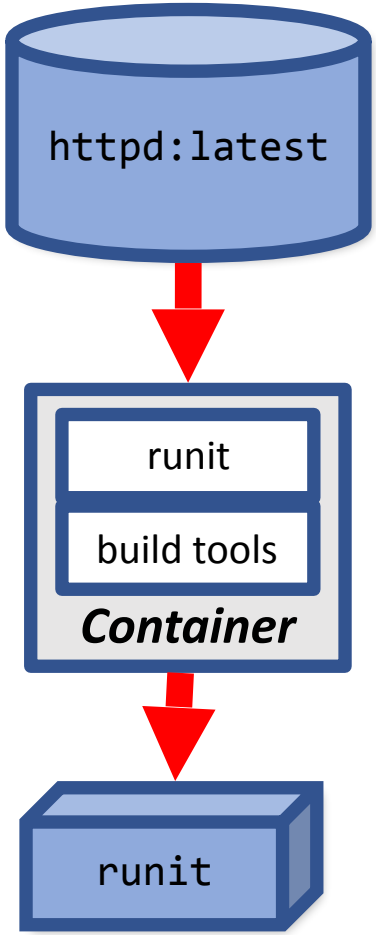
Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`



Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`

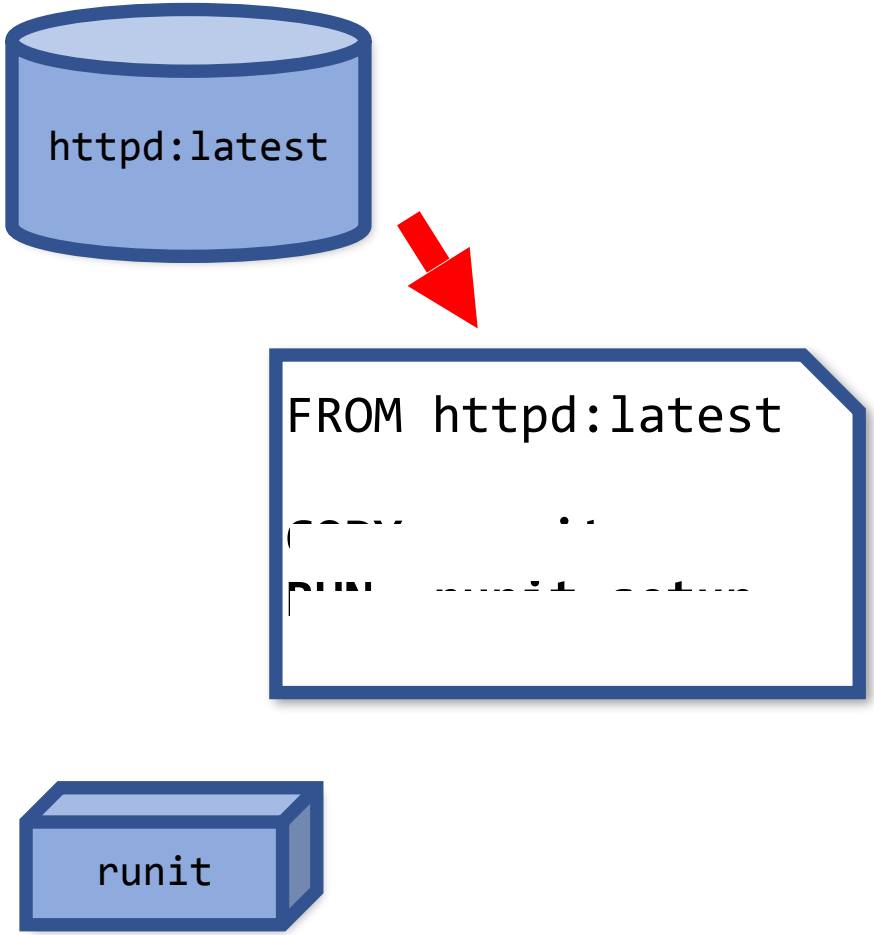


Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`

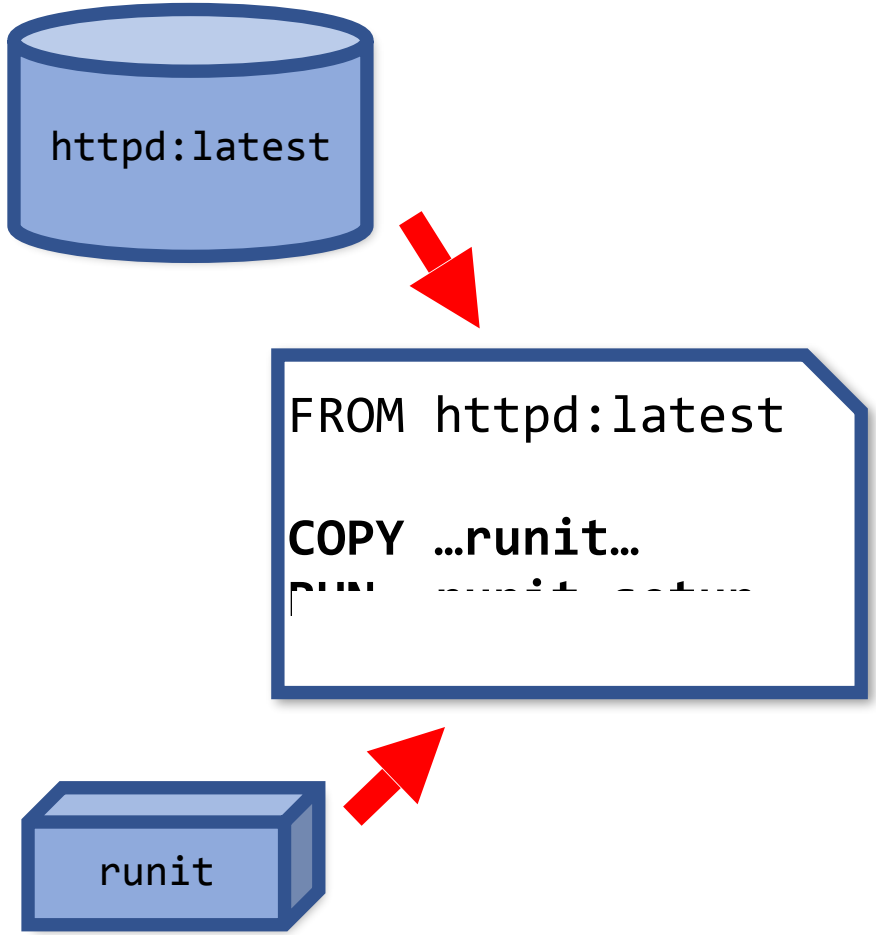
Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`



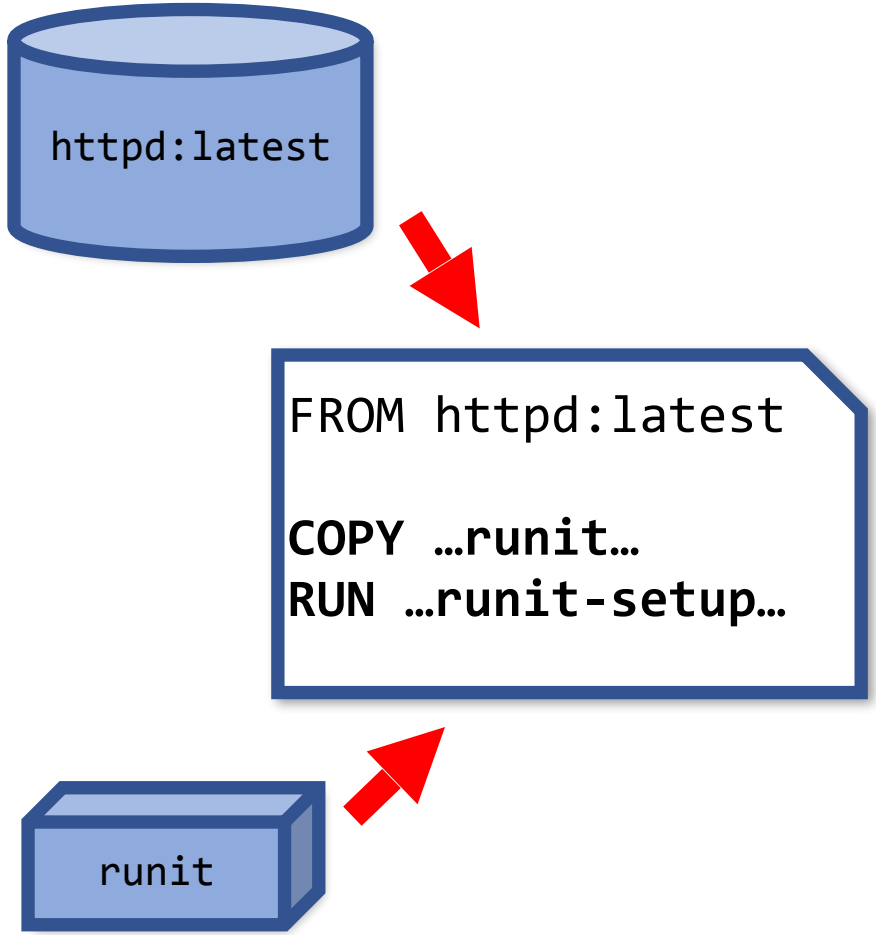
Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`
- add `Dockerfile` instructions to install `runit`



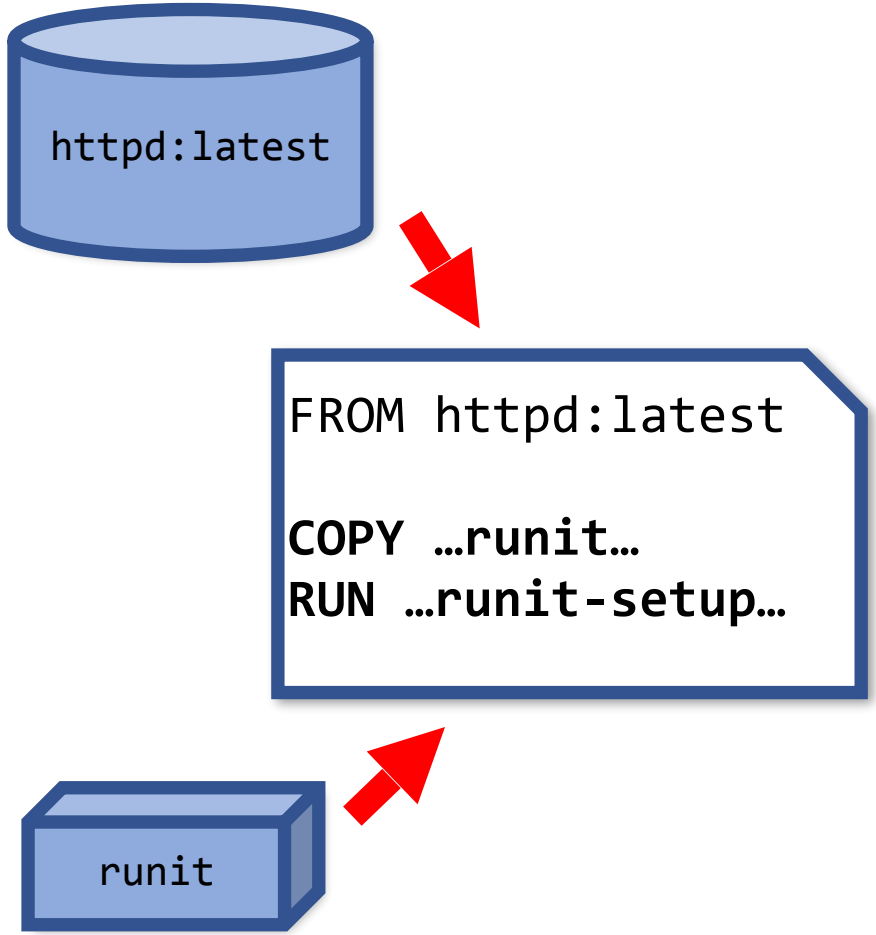
Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`
- add `Dockerfile` instructions to install `runit`
- configure `runit` to run the original `ENTRYPOINT`



Augment the startup

- make temporary container
- add build toolchain
- compile and package `runit`
- add `Dockerfile` instructions to install `runit`
- configure `runit` to run the original `ENTRYPOINT`
- when augmented image is used, set `ENTRYPOINT` to `runit`



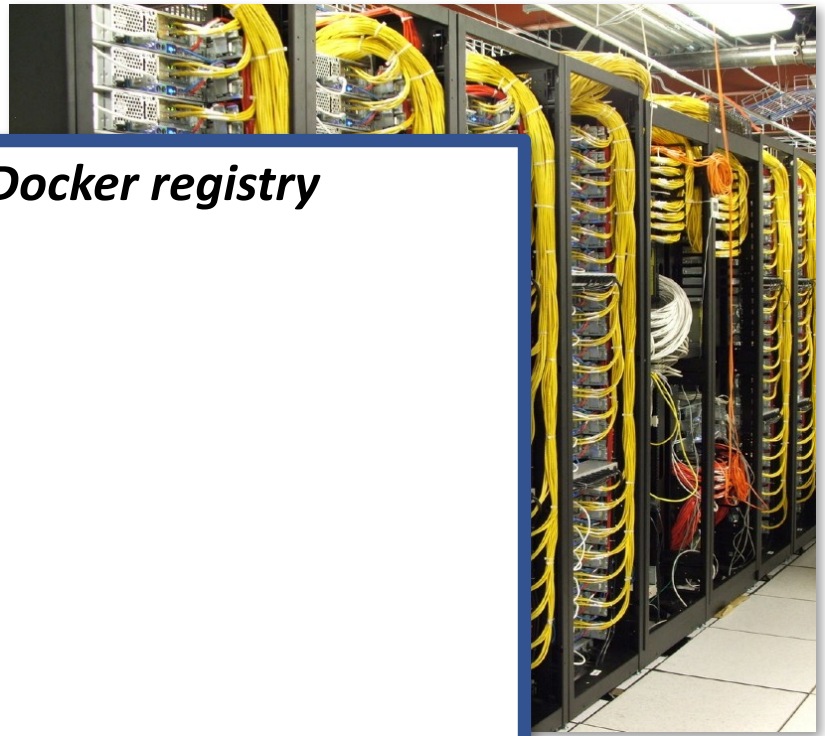
Add the Emulab “client-side” software

- make temporary container
- compile and package Emulab client-side software
- add `Dockerfile` instructions to install the software

- user-selectable levels of augmentation

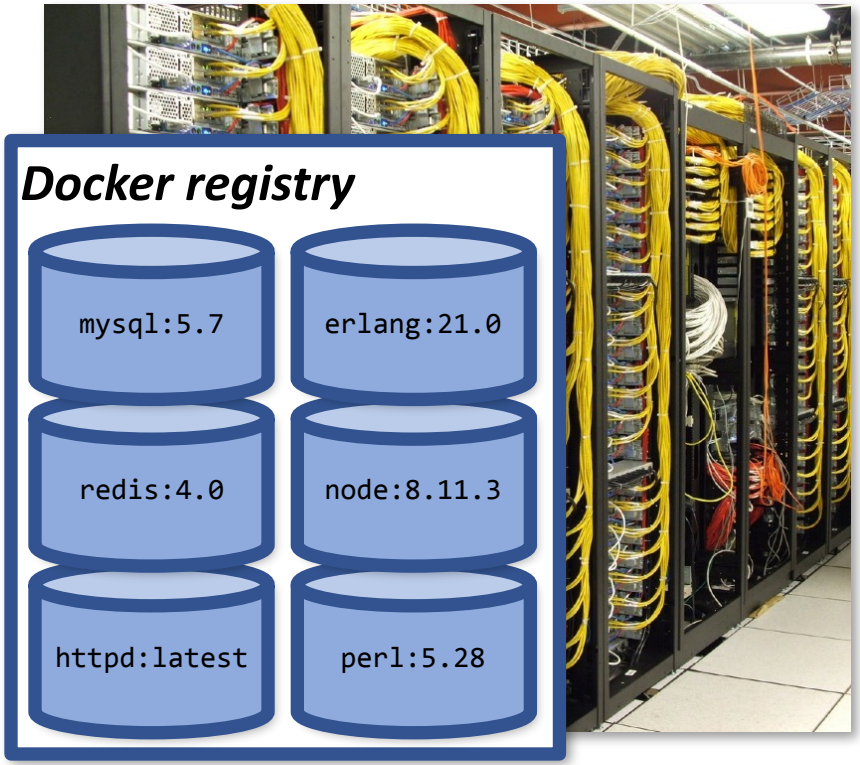
```
FROM httpd:latest

COPY …
RUN …runit-setup…
    && …emulab-setup…
```

Local registry



Docker registry

Local registry



- cache augmented images in a testbed-local Docker registry
- speeds subsequent experiment creation
- integrated with Emulab's user authentication & authorization model

Preserving Emulab's network services

- separate control network
- experiment traffic shaping
- control-network firewalls
- *DNS*

Preserving Emulab's network services

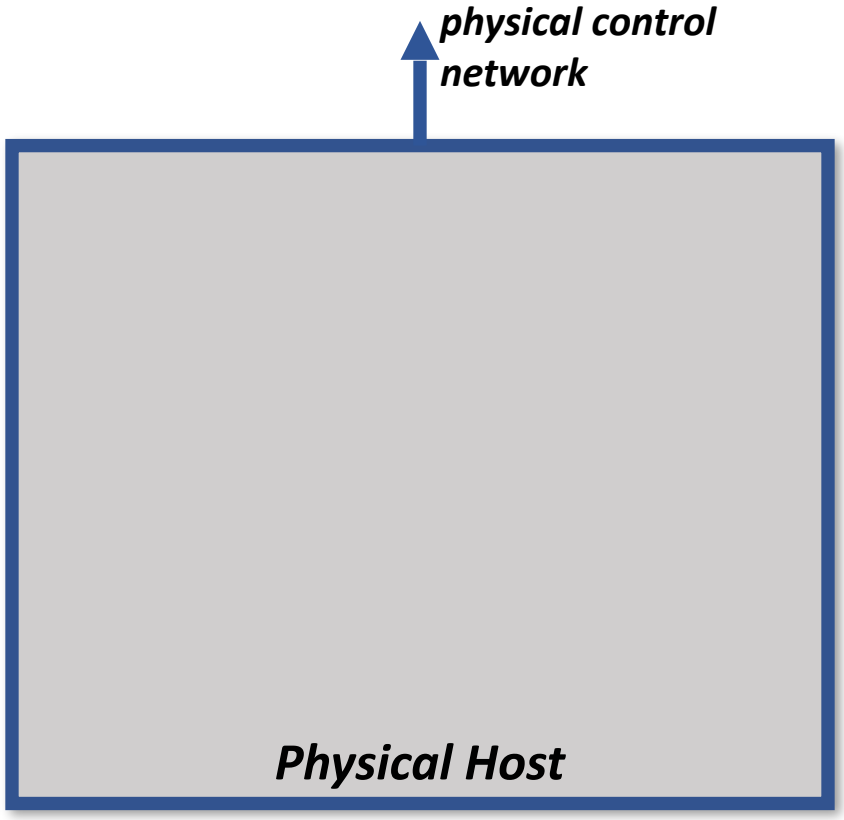
- separate control network
- experiment traffic shaping
- control-network firewalls
- *DNS*

- Docker's Container Network Model (CNM) is **mismatched** to demands of a network testbed
- too abstract
- tries to control too much
- missing features

leverage the physical host

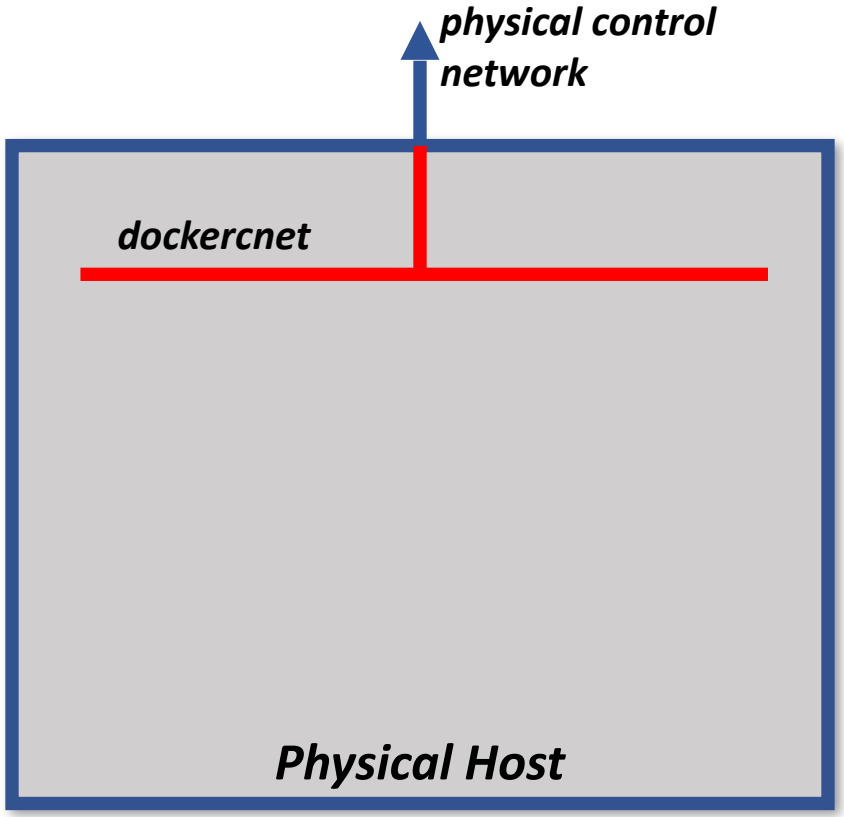
manage network services
on the physical-host side of
containers' virtual network interfaces

Control network



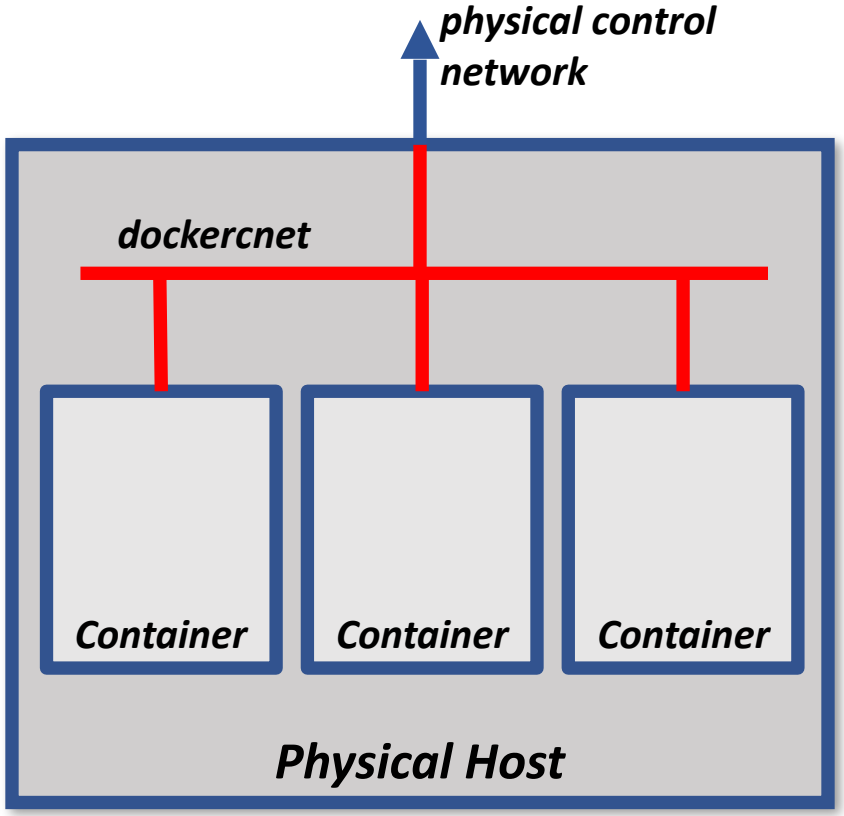
Control network

- at physical-host boot
  - create `dockercnet` virtual network
  - bridge to the physical control network

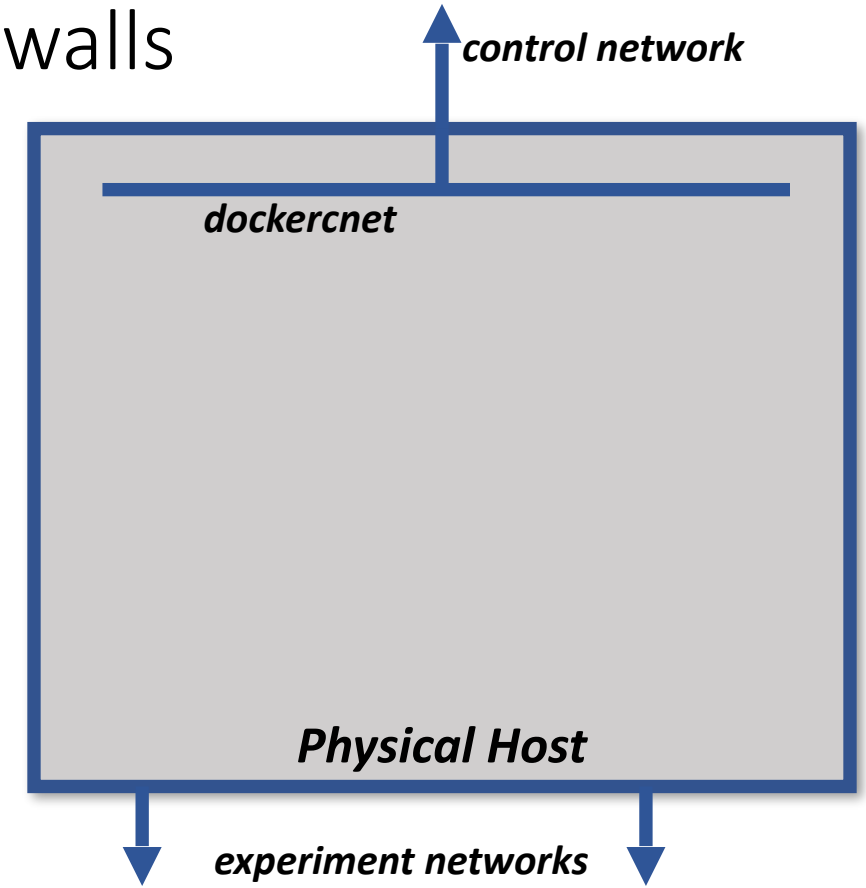


Control network

- at physical-host boot
  - create `dockercnet` virtual network
  - bridge to the physical control network
- at container startup
  - connect to `dockercnet`
  - set up NAT to expose SSH over the physical host's public IP address



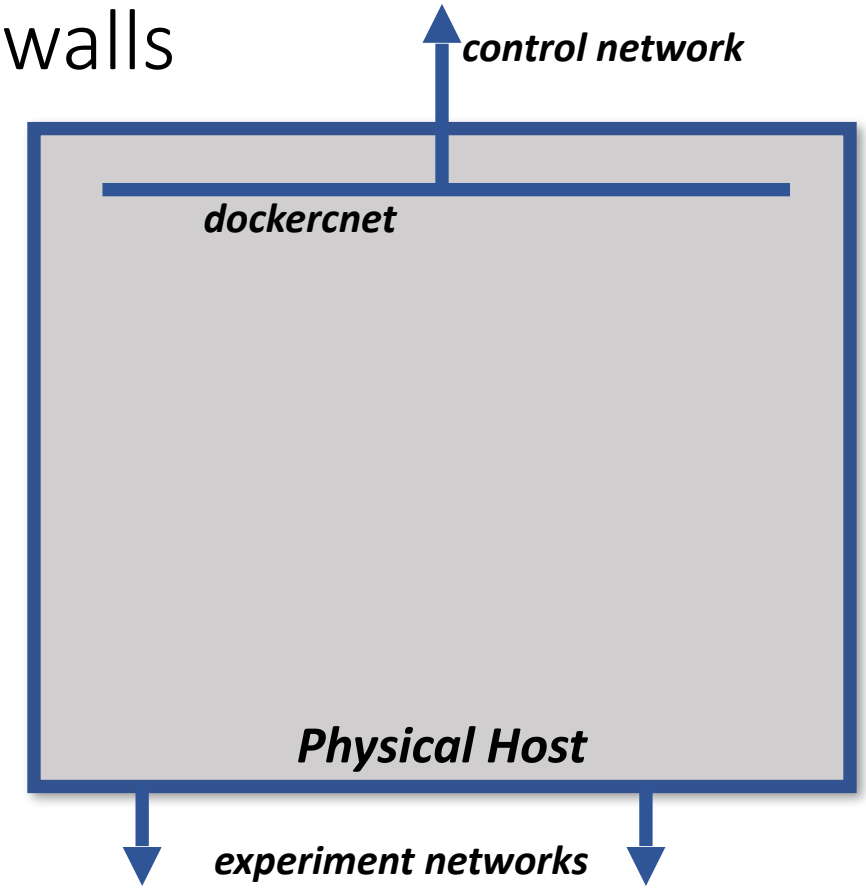
Traffic shaping and firewalls



Traffic shaping and firewalls

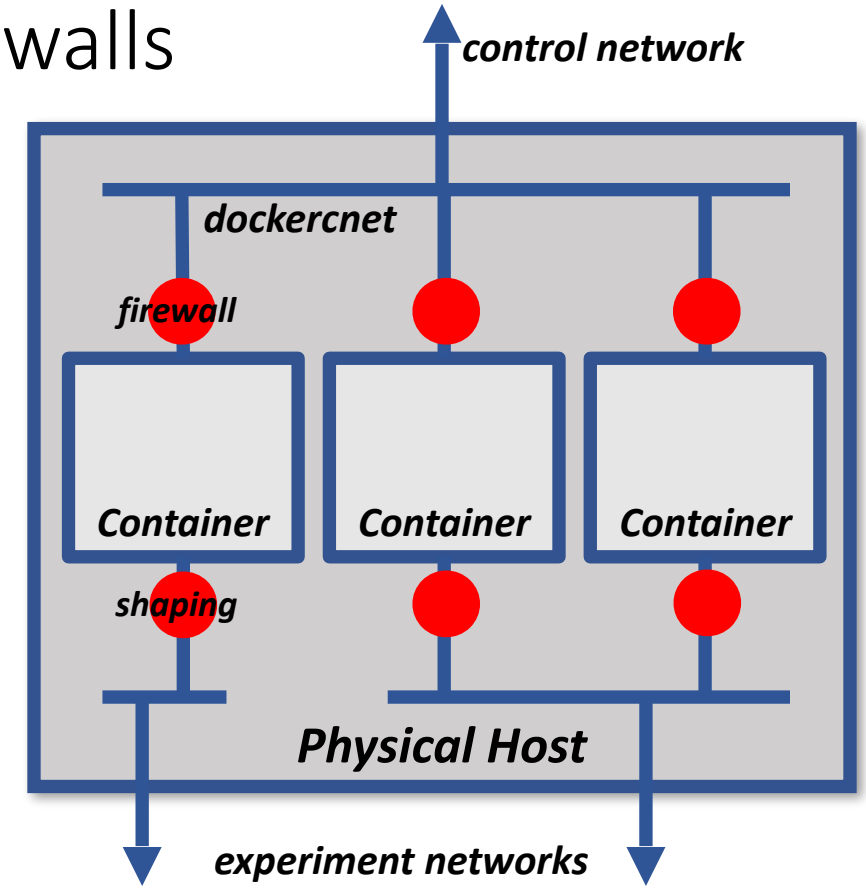
- Emulab subscribes to life-cycle events of each container

- at container startup
  - install `tc` rules for expt.-network traffic shaping
  - install `iptables` rules for control-network firewalling

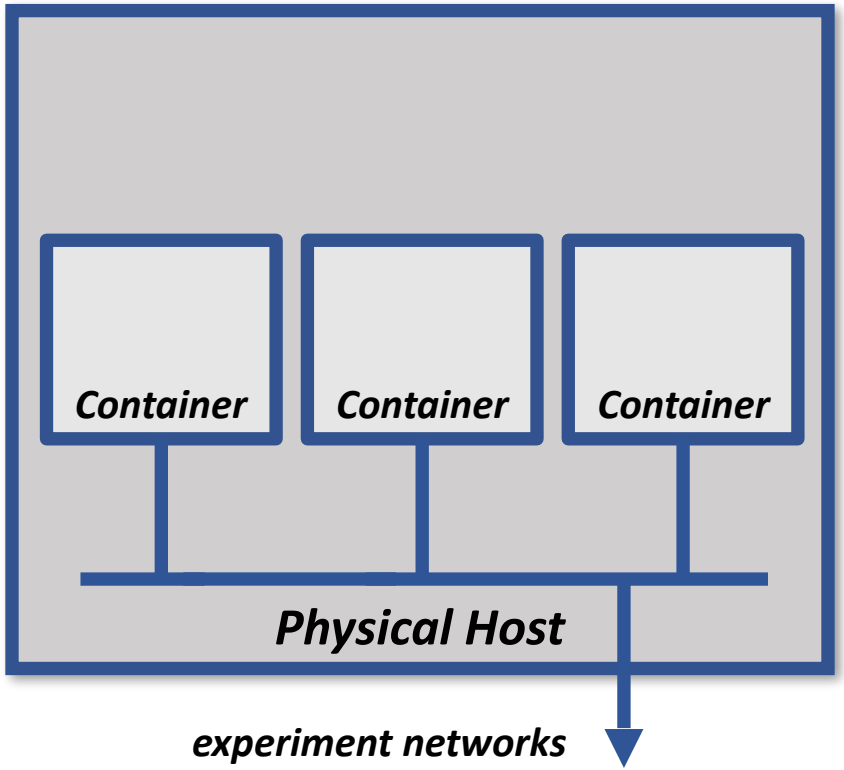


Traffic shaping and firewalls

- Emulab subscribes to life-cycle events of each container
- at container startup
  - install `tc` rules for expt.-network traffic shaping
  - install `iptables` rules for control-network firewalling
- at container shutdown
  - remove the rules

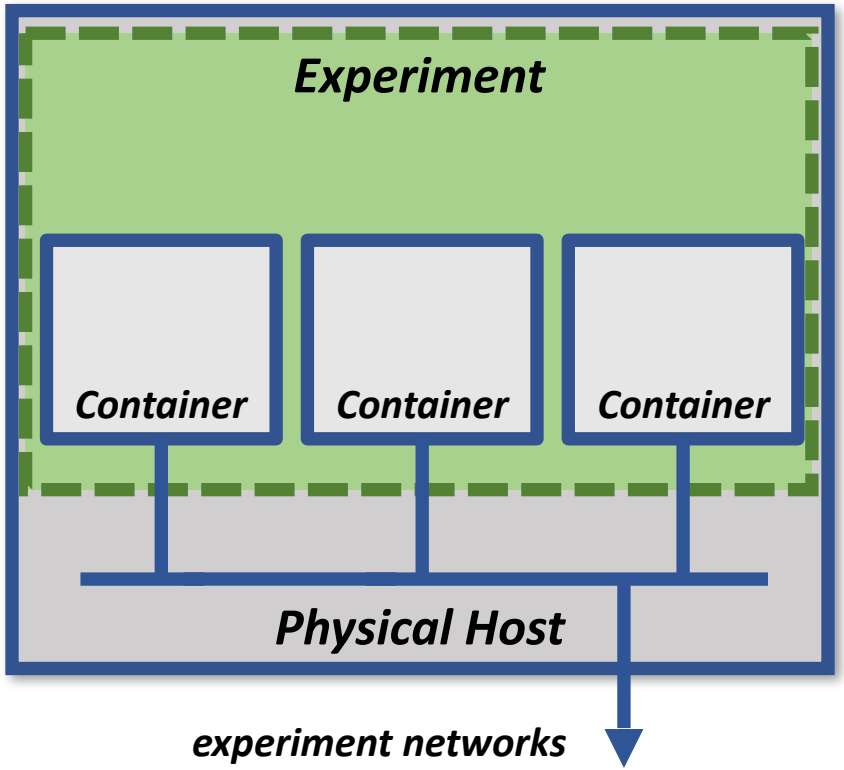


Dedicated and shared modes



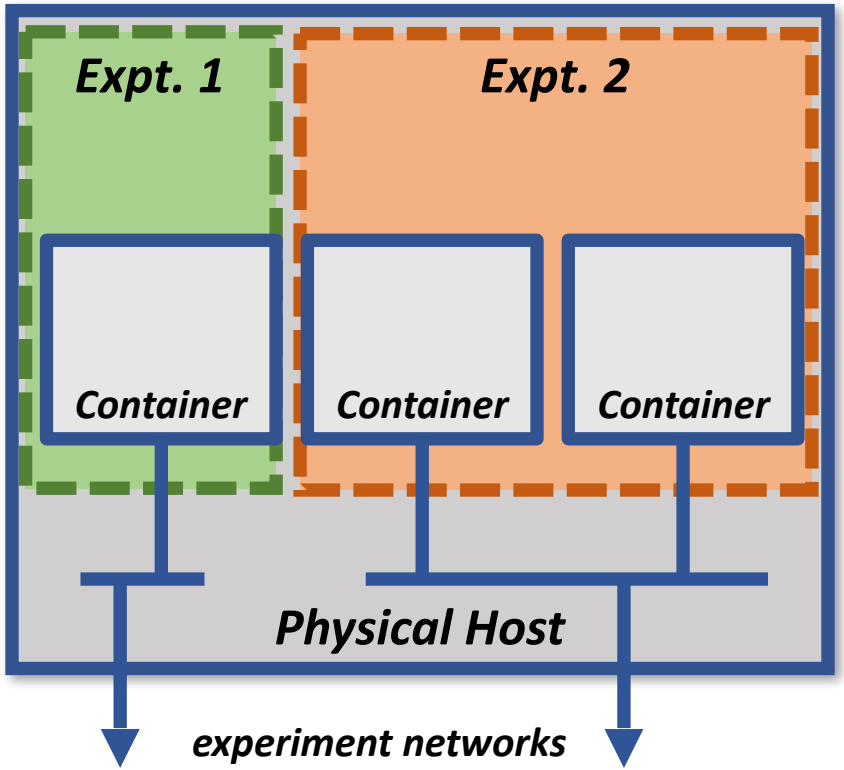
Dedicated and shared modes

- **dedicated**—containers run on physical machine reserved to one experiment



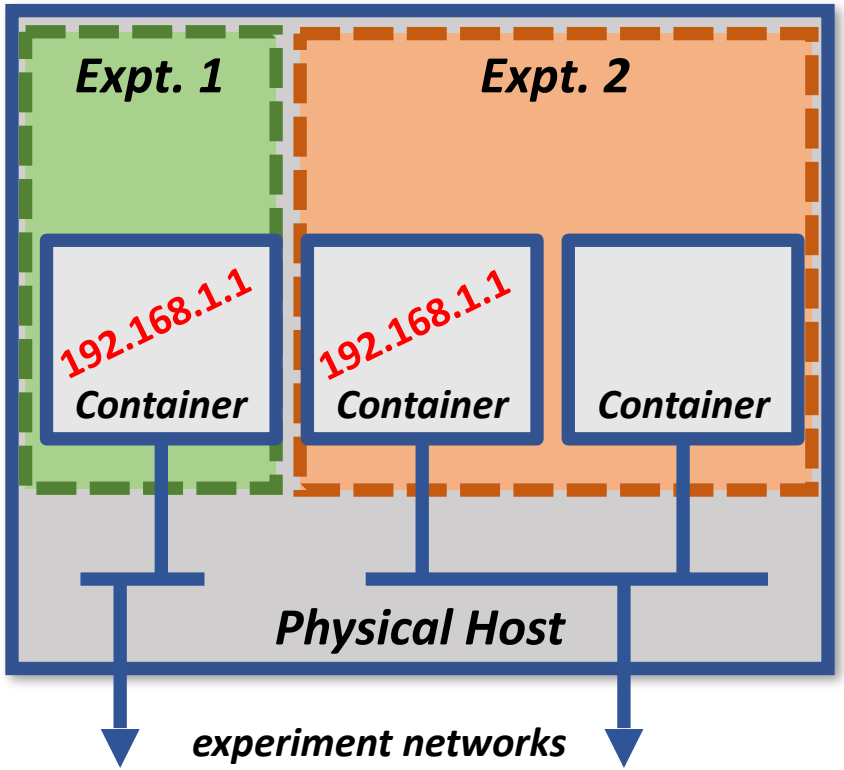
Dedicated and shared modes

- **dedicated**—containers run on physical machine reserved to one experiment
- **shared**—physical machine may host containers from several experiments



Dedicated and shared modes

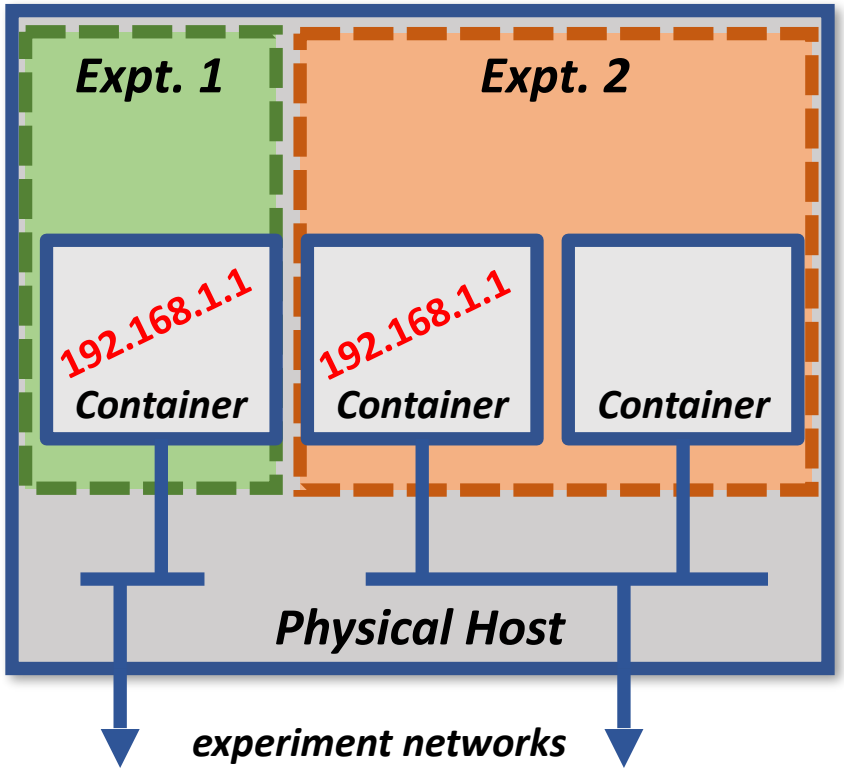
- **dedicated**—containers run on physical machine reserved to one experiment
- **shared**—physical machine may host containers from several experiments



Dedicated and shared modes

- **dedicated**—containers run on physical machine reserved to one experiment
- **shared**—physical machine may host containers from several experiments

- we modified Docker to support multiple, isolated layer 2 nets on a single physical host



Implemented & deployed

- supported OSes
  - Alpine Linux 3.6, 3.7, 3.8
  - CentOS 7
  - Debian 8, 9, sid
  - Ubuntu 14.04, 16.04, 18.04

- registries at

evaluation

- 60 most popular images from Docker Hub
- *four research Docker images*
- *time to augment Docker images*
- time to create large experiments

Category	Docker Images
Linux distro	alpine, centos, debian, ubuntu, amazonlinux, busybox, fedora
Debian	buildpack-deps, cassandra, chronograf, drupal, elasticsearch, ghost, golang, gradle, groovy, haproxy, httpd, influxdb, java, jenkins, jruby, kibana, logstash, mariadb, maven, memcached, mongo, mysql, nextcloud, nginx, node, openjdk, owncloud, percona, perl, php, postgres, python, rabbitmq, redis, rethinkdb, rocket.chat, ruby, sentry, solr, sonarqube, tomcat, wordpress, telegraf
Alpine	consul, docker, kong, neo4j, vault, registry
Scratch	hello-world, nats, swarm, traefik

Category	Docker Images
Linux distro	alpine, centos, debian, ubuntu, amazonlinux, busybox, fedora
Debian	buildpack-deps, cassandra, chronograf, drupal, elasticsearch, ghost, golang, gradle, groovy, haproxy, httpd, influxdb, java, jenkins, jruby, kibana, logstash, mariadb, maven, memcached, mongo, mysql, nextcloud, nginx, node, openjdk, owncloud, percona, perl, php, postgres, python, rabbitmq, redis, rethinkdb, rocket.chat, ruby, sentry, solr, sonarqube, tomcat, wordpress, telegraf
Alpine	consul, docker, kong, neo4j, vault, registry
Scratch	hello-world, nats, swarm, traefik

fully supported
partially supported
~~not supported~~

Category	Docker Images
Linux distro	alpine, centos, debian, ubuntu, ~~amazonlinux~~, ~~busybox~~, ~~fedora~~
Debian	buildpack-deps, cassandra, chronograf, drupal, elasticsearch, ghost, golang, gradle, groovy, haproxy, httpd, influxdb, java, jenkins, jruby, kibana, logstash, mariadb, maven, memcached, mongo, mysql, nextcloud, nginx, node, openjdk, owncloud, percona, perl, php, postgres, python, rabbitmq, redis, rethinkdb, rocket.chat, ruby, sentry, solr, sonarqube, tomcat, wordpress, *telegraf*
Alpine	consul, docker, kong, neo4j, vault, ~~registry~~
Scratch	~~hello-world~~, ~~nats~~, ~~swarm~~, ~~traefik~~

fully supported
partially supported
~~not supported~~

Category	Docker Images
Linux distro	alpine, centos, debian, ubuntu, ~~amazonlinux~~, ~~busybox~~, ~~fedora~~
Debian	buildpack-deps, cassandra, chronograf, drupal, elasticsearch, ghost, golang, haproxy, httpd, influxdb, jav, kibana, logstash, mariadb, r, mongo, mysql, nextcloud, n, owncloud, percona, perl, ph, rabbitmq, redis, rethinkdb, sentry, solr, sonarqube, tomcat, wordpress, *telegraf*
Alpine	consul, docker, kong, neo4j, vault, ~~registry~~
Scratch	~~hello-world~~, ~~nats~~, ~~swarm~~, ~~traefik~~

fully supported
partially supported
~~not supported~~

Emulab automatically adapted **52/60** images into the testbed environment and instantiated containers from them.

Scalability

- **create large experiments with Docker containers**

- in each trial
  - 200 containers per physical host
  - each container runs augmented `ubuntu:14.04` image from testbed's local registry
  - all containers attached to a LAN

  - physical hosts: CloudLab xl170 nodes running Ubuntu 16.04

Scalability

- **create large experiments with Docker containers**

- in each trial
  - 200 containers per physical host
  - each container runs augmented `ubuntu:14.04` image from testbed's local registry
  - all containers attached to a LAN

  - physical hosts: CloudLab xl170 nodes running Ubuntu 16.04

- 1–25 physical hosts
  - yielding **200–5,000 containers**

Scalability

- **create large experiments with Docker containers**

- in each trial
  - 200 containers per physical host
  - each container runs augmented `ubuntu:14.04` image from testbed's local registry
  - all containers attached to a LAN

  - physical hosts: CloudLab xl170 nodes running Ubuntu 16.04

- 1–25 physical hosts
  - yielding **200–5,000 containers**

- measure
  - elapsed time to first container
  - avg. creation time for each container after the first
  - elapsed time to create all containers on each physical host
  - elapsed time to create full expt.

Scalability

- **create large experiments with Docker containers**

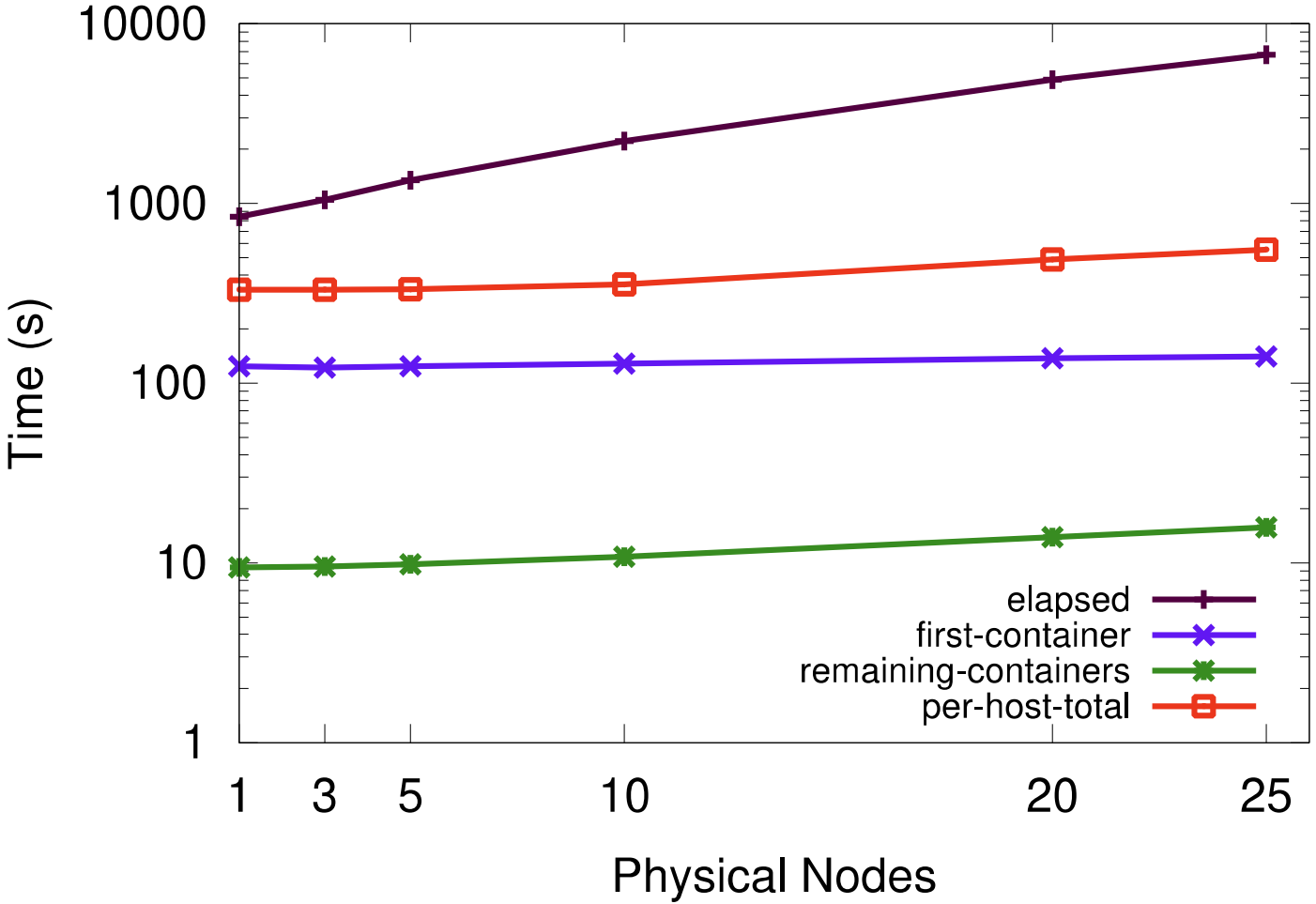
- in each trial
  - 200 containers per physical host
  - each container runs augmented `ubuntu:14.04` image from testbed's local registry
  - all containers attached to a LAN

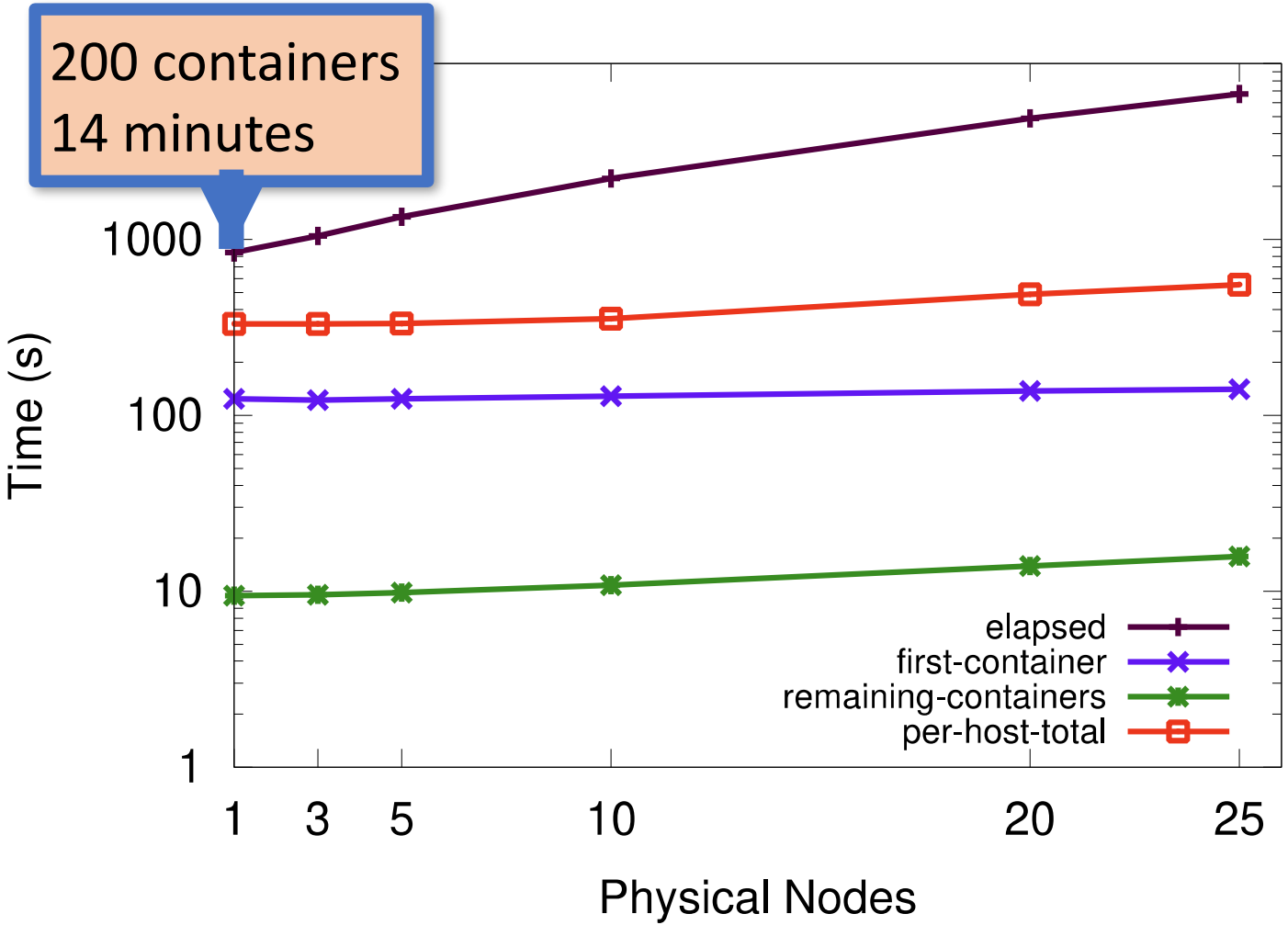
  - physical hosts: CloudLab xl170 nodes running Ubuntu 16.04

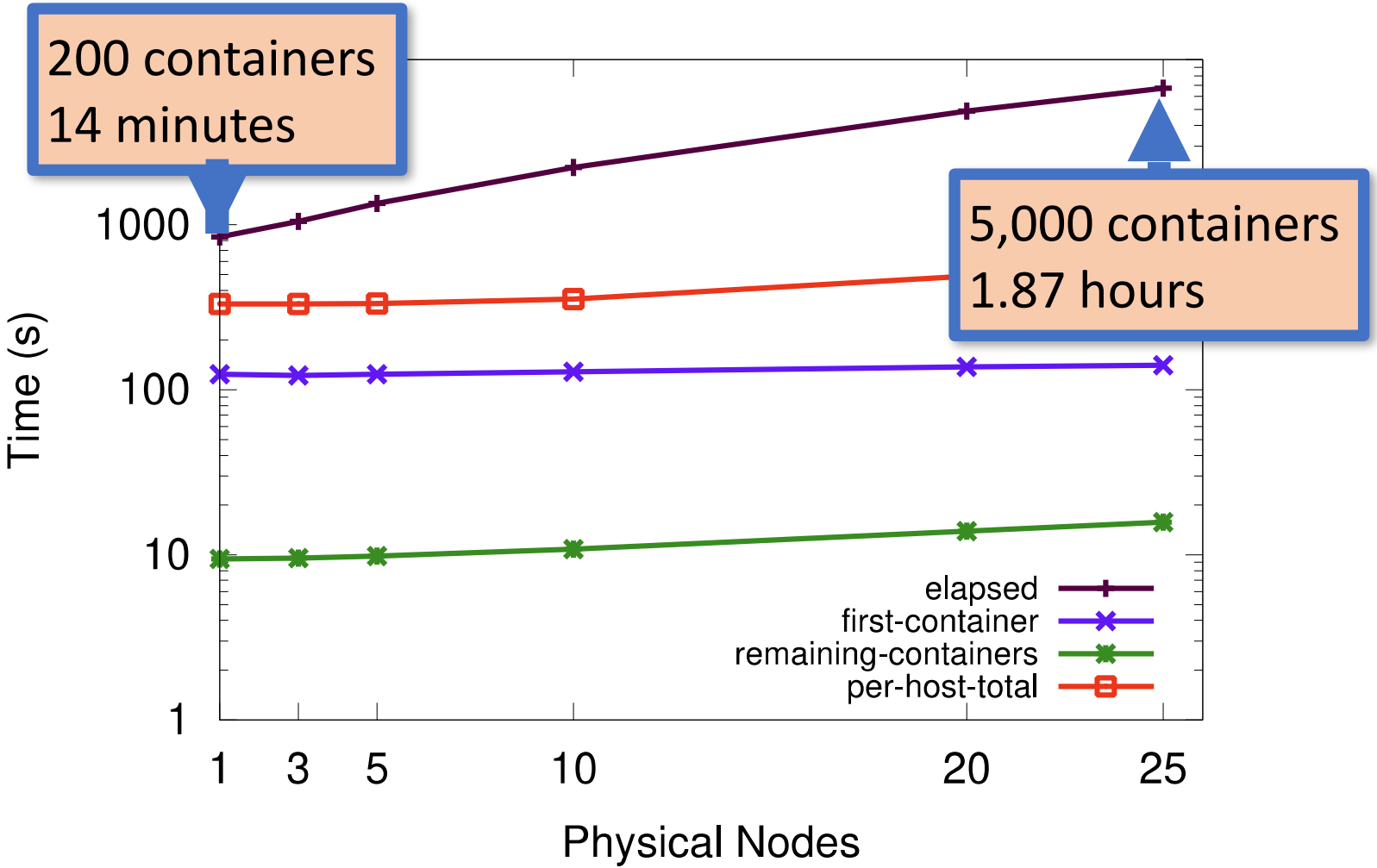
- 1–25 physical hosts
  - yielding **200–5,000 containers**

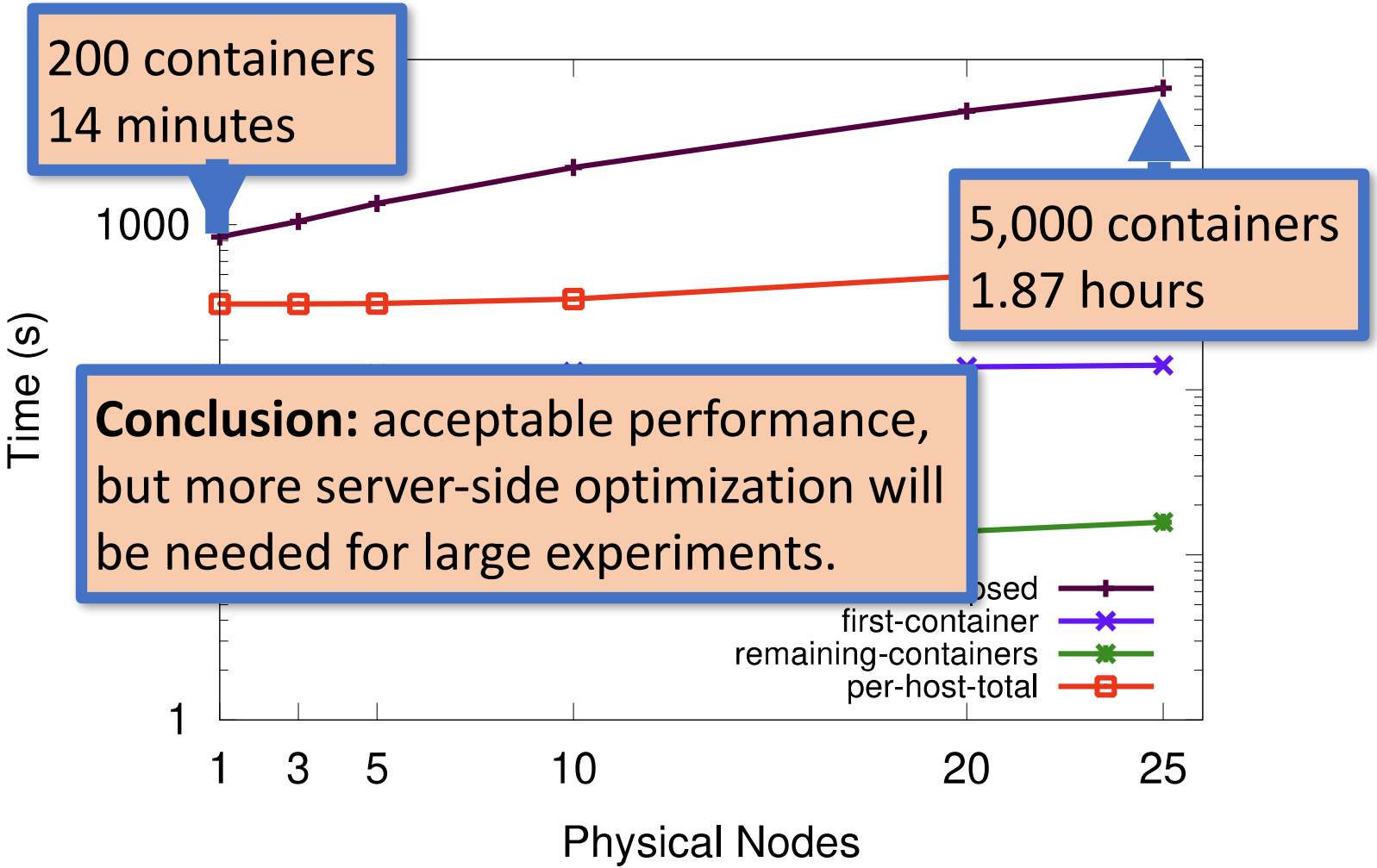
- measure
  - elapsed time to first container
  - avg. creation time for each container after the first
  - elapsed time to create all containers on each physical host
  - elapsed time to create full expt.

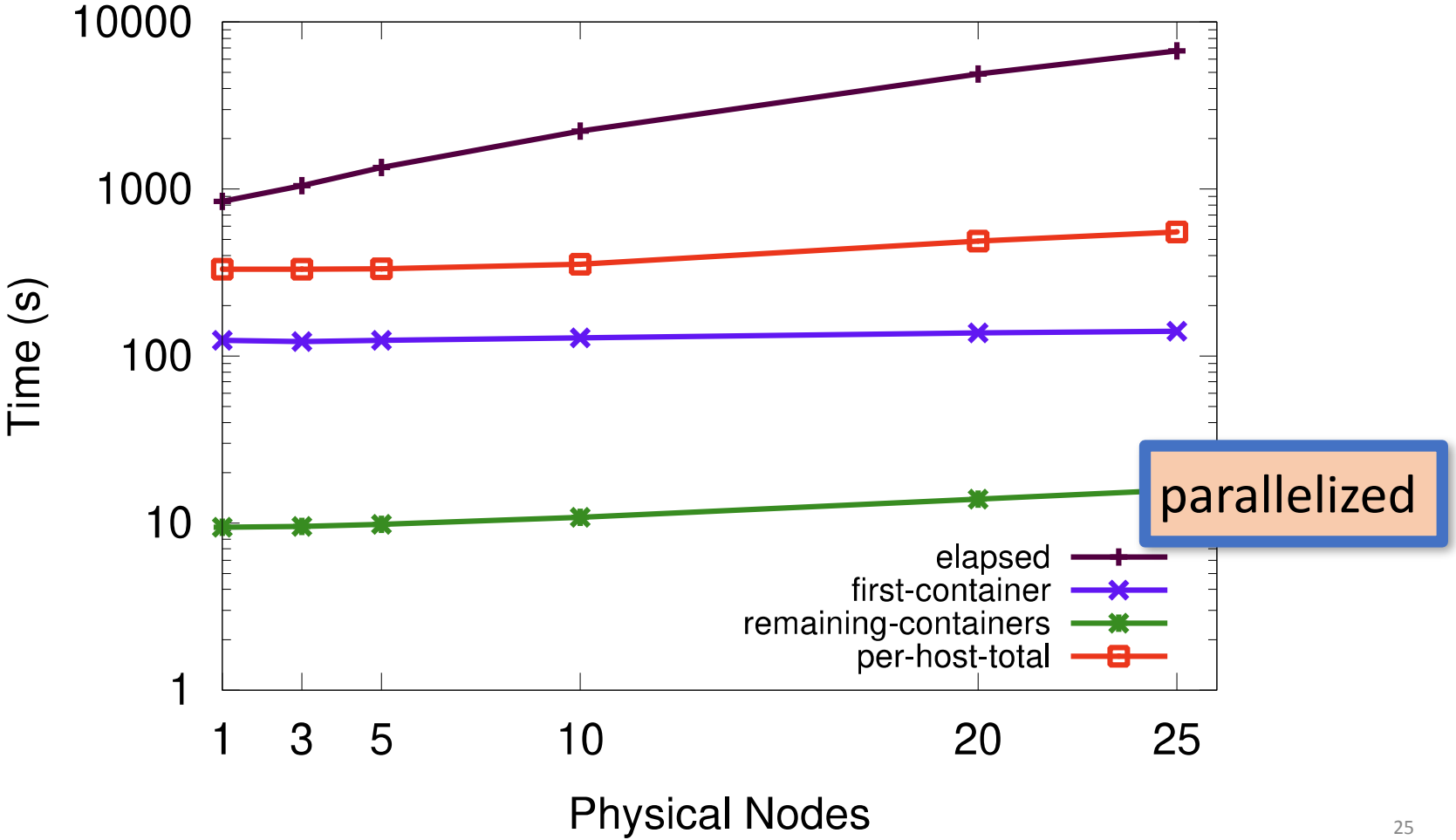
- repeat each trial 3×, report avgs.

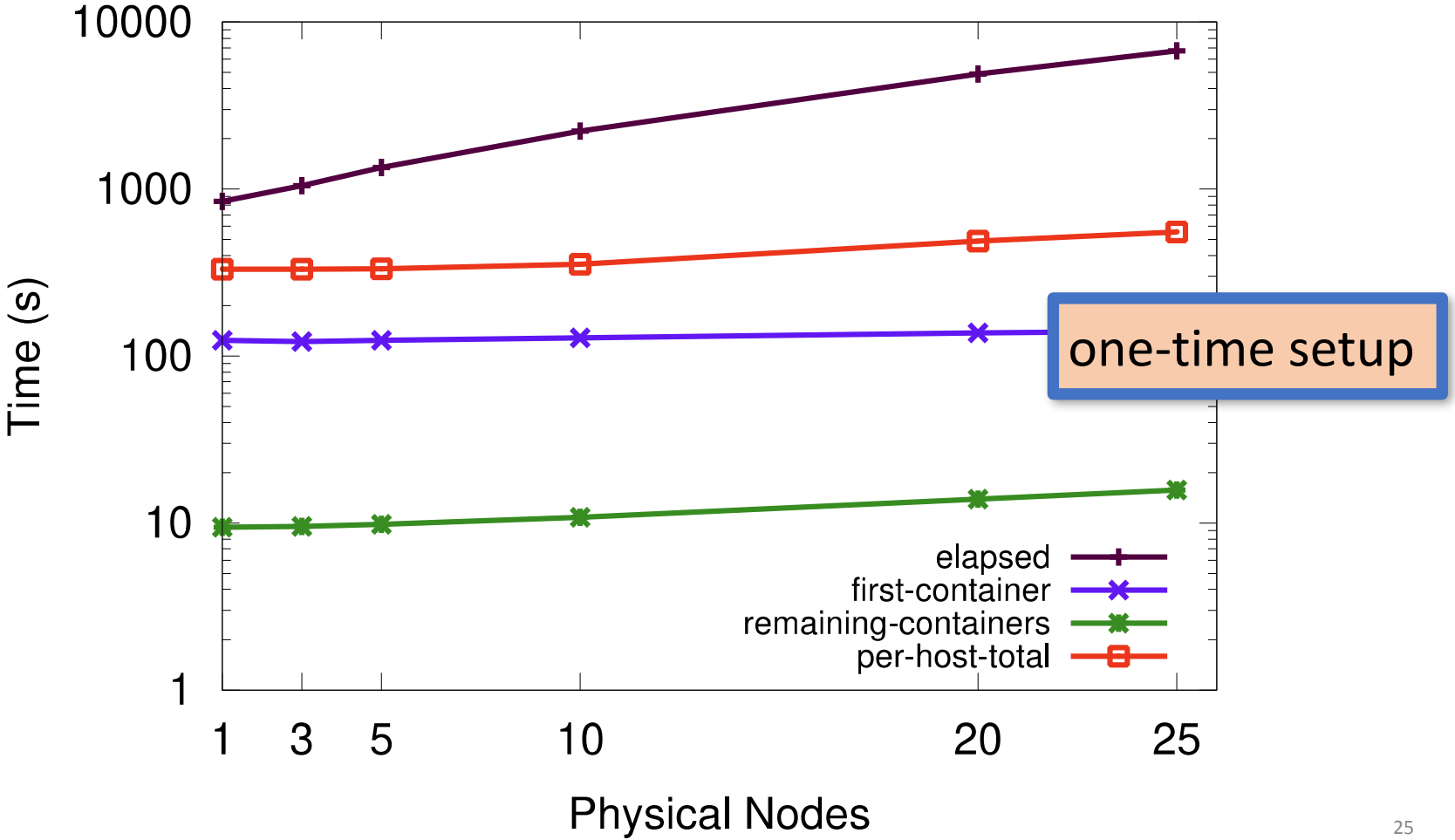

10000
1000
100
10
1
Time (s)
1
3
5
10
20
25
Physical Nodes
elapsed
first-container
remaining-containers
per-host-total


200 containers
14 minutes
1000
100
10
1
Time (s)
1
3
5
10
20
25
Physical Nodes
elapsed
first-container
remaining-containers
per-host-total


200 containers
14 minutes
5,000 containers
1.87 hours
1000
100
10
1
Time (s)
1
3
5
10
20
25
Physical Nodes
elapsed
first-container
remaining-containers
per-host-total


200 containers
14 minutes
5,000 containers
1.87 hours
Conclusion: acceptable performance, but more server-side optimization will be needed for large experiments.
1000
1
Time (s)
1
3
5
10
20
25
Physical Nodes
osed
first-container
remaining-containers
per-host-total


10000
1000
100
10
1
Time (s)
1
3
5
10
20
25
Physical Nodes
elapsed
first-container
remaining-containers
per-host-total
parallelized


10000
1000
100
10
1
Time (s)
1
3
5
10
20
25
Physical Nodes
elapsed
first-container
remaining-containers
per-host-total
one-time setup

Conclusion

- **Emulab + Docker "just works"**
  - experimenter services—automatic augmentation
  - network services—physical host control & minor Docker mods
- supports existing Docker images
- promotes artifact portability
- promotes research repeatability
- **available in Emulab-based testbeds now!**

Eric Eide
www.cs.utah.edu/~eeide/
email: eeide@cs.utah.edu
Twitter: @eeide