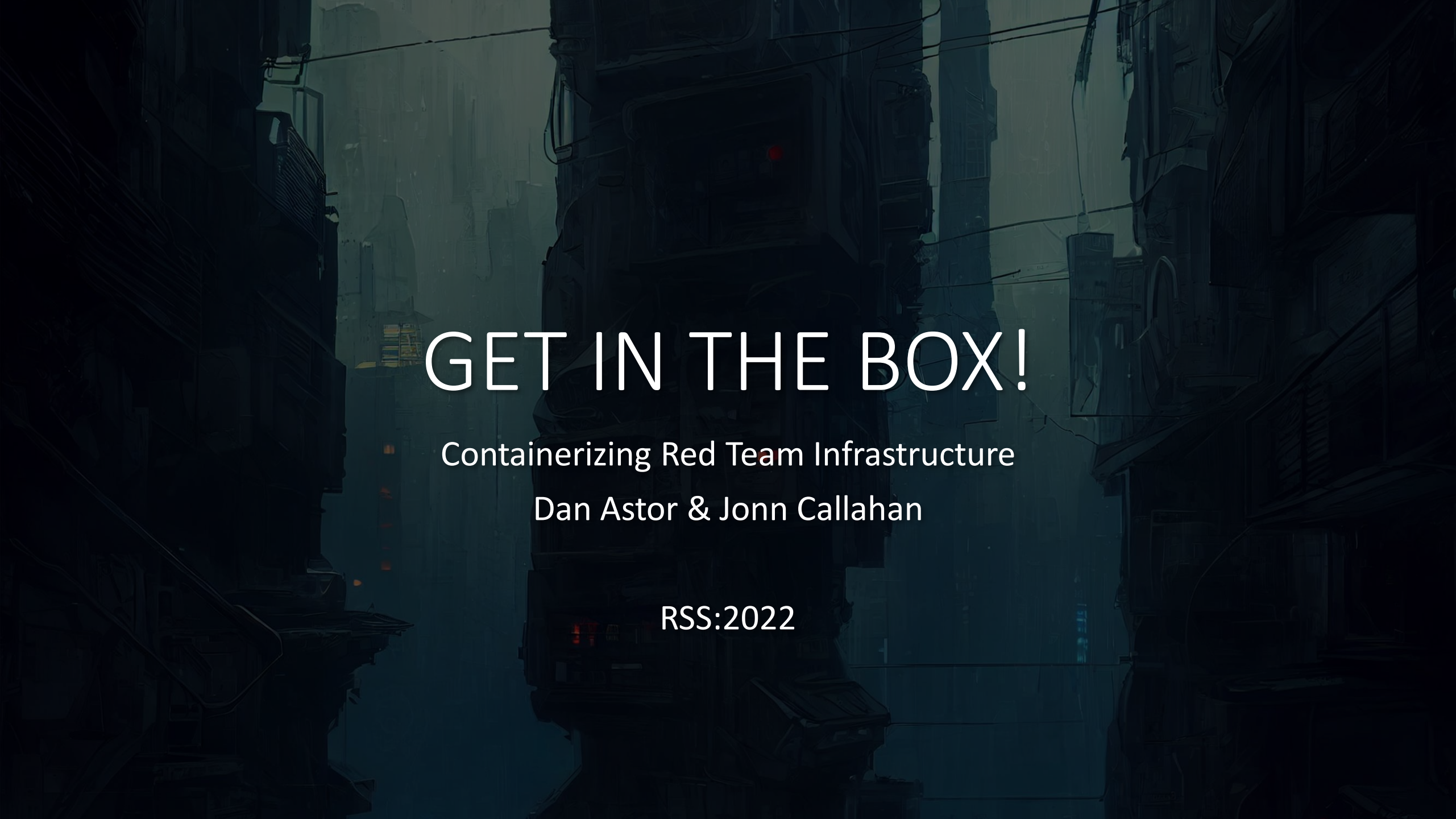


GET IN THE BOX!

Containerizing Red Team Infrastructure

Dan Astor & Jonn Callahan

RSS:2022

Example Implementation

https://github.com/SecurityRiskAdvisors/GetInTheBox

Agenda

Introduction

Tooling Overview

Common Industry Practices

Terraform & Docker & Swarm

Traefik & Routing

Monitoring

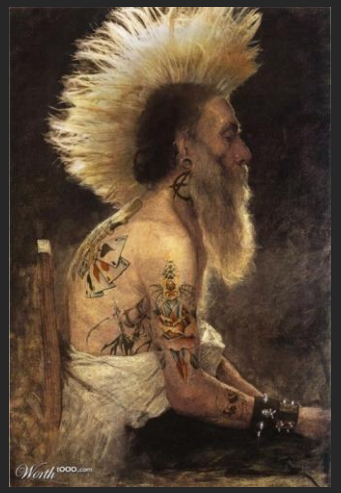
Closing Thoughts

Questions

About Us

Jonn Callahan (@atticuss)

- App & cloudsec background
- Enjoys automating things
- Metalhead



Dan Astor (@illegitimateDA)

- Pentesting & Red Team
- Enjoys breaking things
- Not as metal as Jonn

SecurityRisk ADVISORS

Tool and Concept Overview

Infrastructure as Code (IaC)

Manage and **provision infrastructure** through code and configuration files vs manual processes

Deployment consistency across environments as it deploys the same plan/config each time

Reduces time, manual processes, and costly mistakes

Infrastructure provisioning

Reach a desired state through infrastructure deployment

Common Examples:

- Create virtual compute instances
- Create security groups
- Create virtual private clouds

```
# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a Virtual Private Cloud
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}
```

Configuration Management

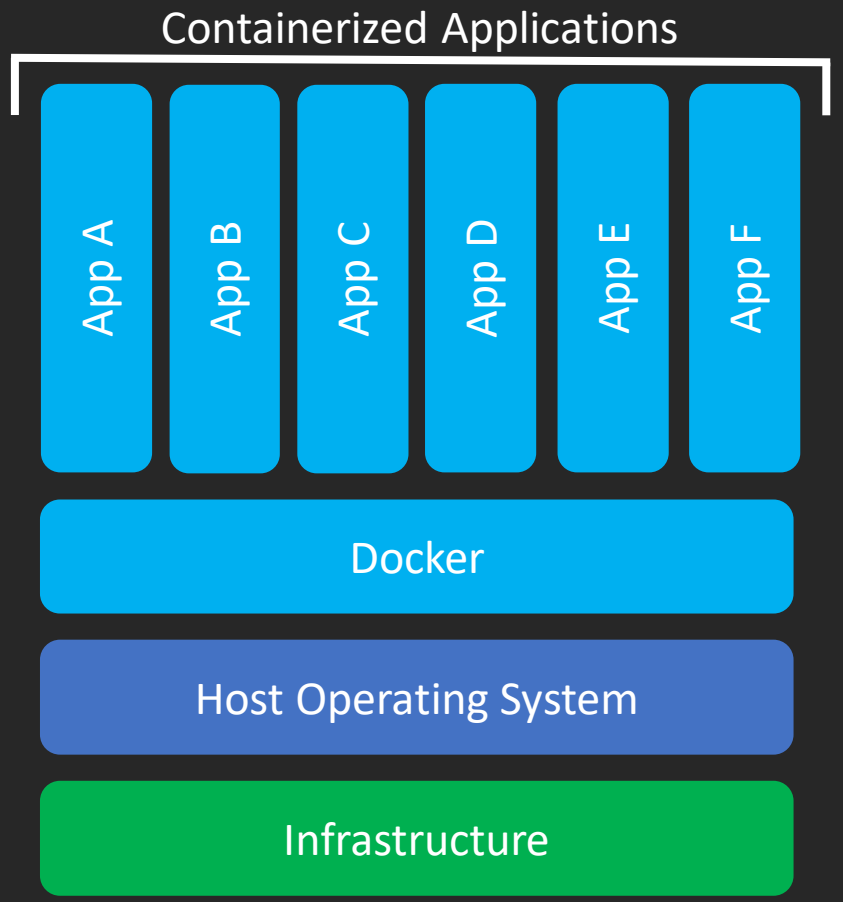
Reach a desired state through OS level changes

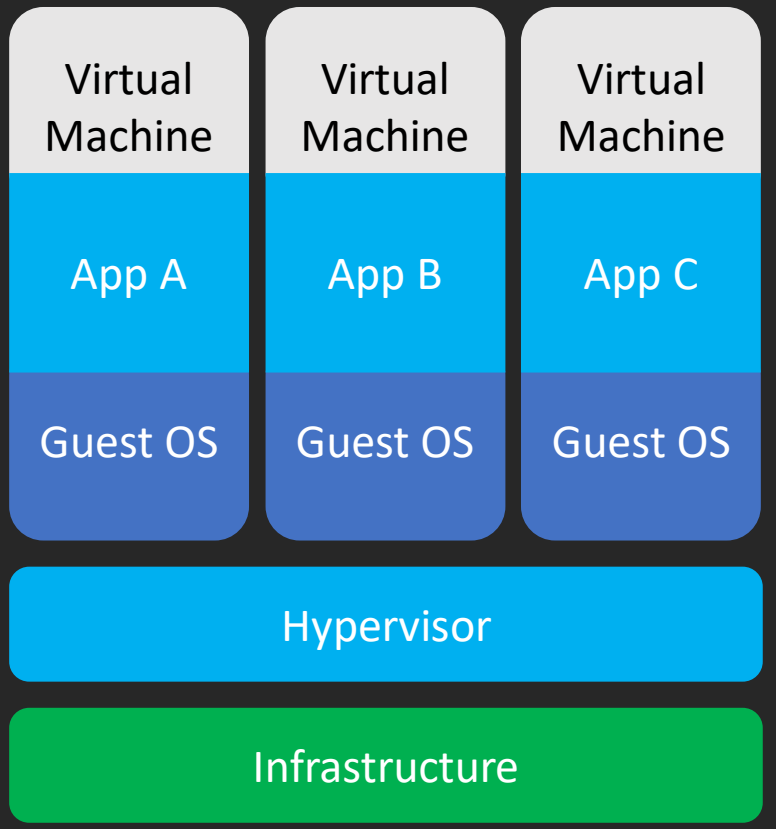
Common Examples:

- Install packages
- Configure a web server
- Update configuration files

```
---
- name: Install a list of packages
  ansible.builtin.apt:
    pkg:
    - nginx
    - nginx-extras

- name: Copy config file
  ansible.builtin.copy:
    src: /srv/myfiles/nginx.conf
    dest: /etc/nginx/nginx.conf
```

Containers


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure


Virtual Machine
Virtual Machine
Virtual Machine
App A
App B
App C
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure

Notable DevOps Tools

Infrastructure Provisioning
- Terraform
- Cloudformation

Configuration Management
- Ansible
- Chef
- Puppet
- SaltStack

Containers
- Docker

Common Industry Practices

Red Team Infrastructure Terminology

C2 Server: The command and control (C2) system that sends and receives tasks, files, and data from implants.

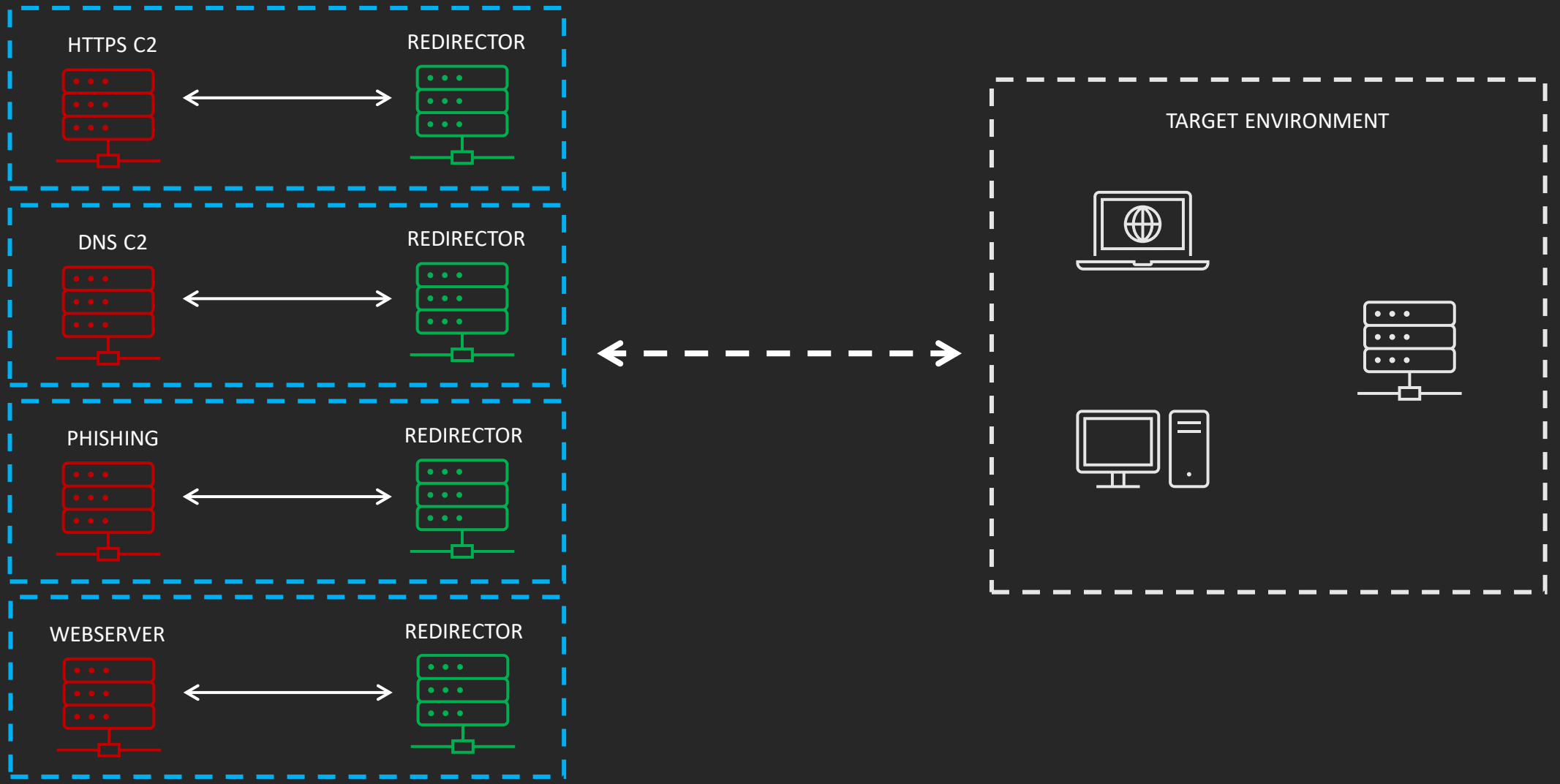
C2 Channel: The communication channel or protocol used between implant and C2 server (e.g., HTTPS, DNS).

C2 Profile: A defined format and structure for C2 traffic to use (e.g., URI paths, agents, headers, etc.).

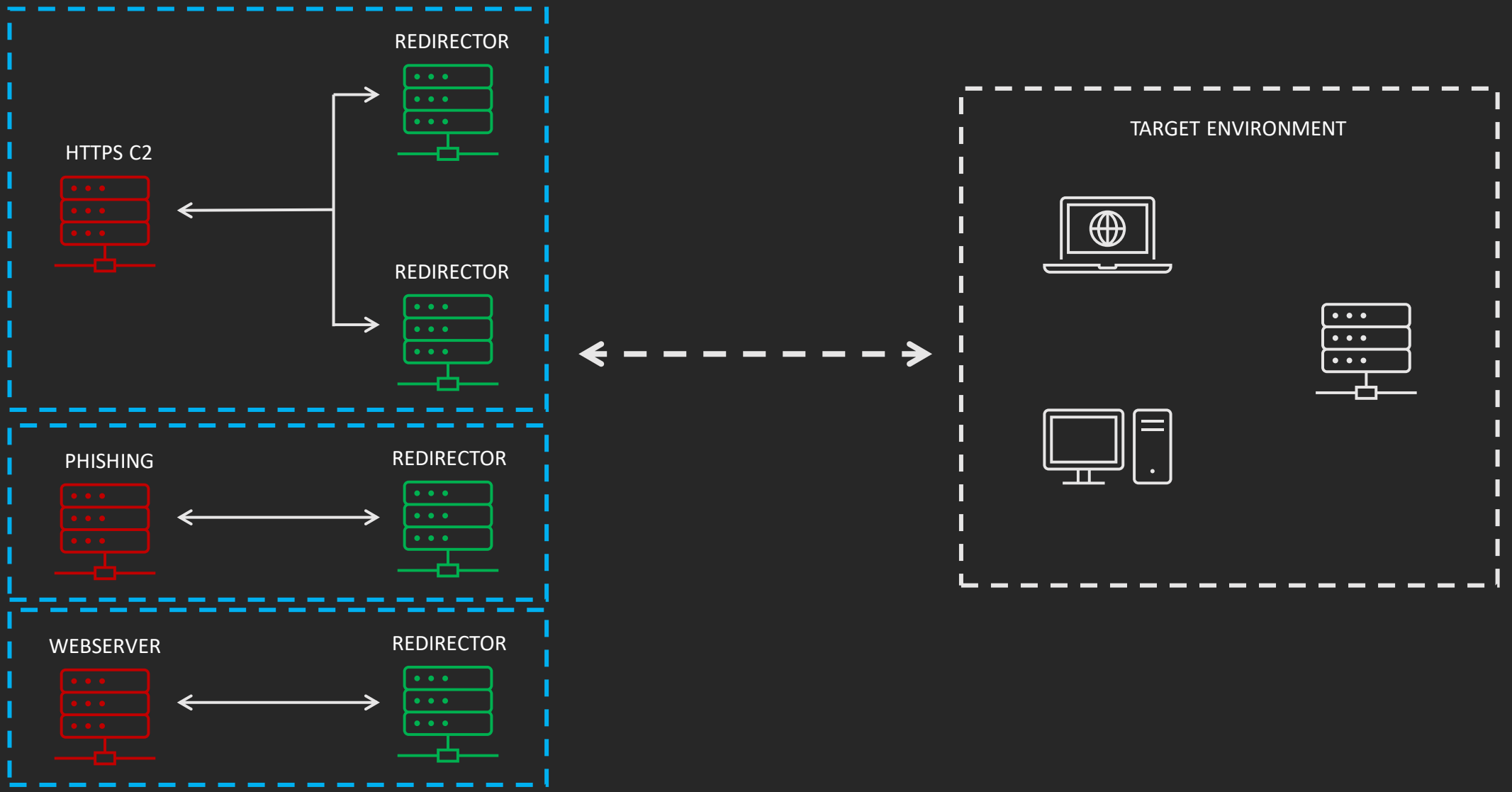
Implant: Piece of code that is created to run on a victim host that connects back to a C2 server.

Redirector: Protects your infrastructure by filtering and routing traffic.

Example Deployment



Example Deployment



Previous Deployment Strategy

Create individual systems for each service (e.g., Redirector, C2, etc.)

1. Define Terraform plan
2. Deploy instances with Terraform
3. Configure instances with Ansible
4. Perform engagement
5. Destroy instances with Terraform

Frustrations & Pitfalls

Higher costs with many single use systems

Maintenance of multiple roles/providers

Ansible OS level configurations

Long build times

Architecture inflexibility

Certificate & domain management

Notable Work

https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki

https://rastamouse.me/infrastructure-as-code-terraform-ansible/

https://github.com/Coalfire-Research/Red-Baron

https://www.mdsec.co.uk/2020/02/testing-your-redteam-infrastructure/

https://github.com/warhorse/warhorse

https://byt3bl33d3r.substack.com/p/taking-the-pain-out-of-c2-infrastructure-3c4

Terraform & Docker & Swarm

Primary Goal: Ease-of-Use

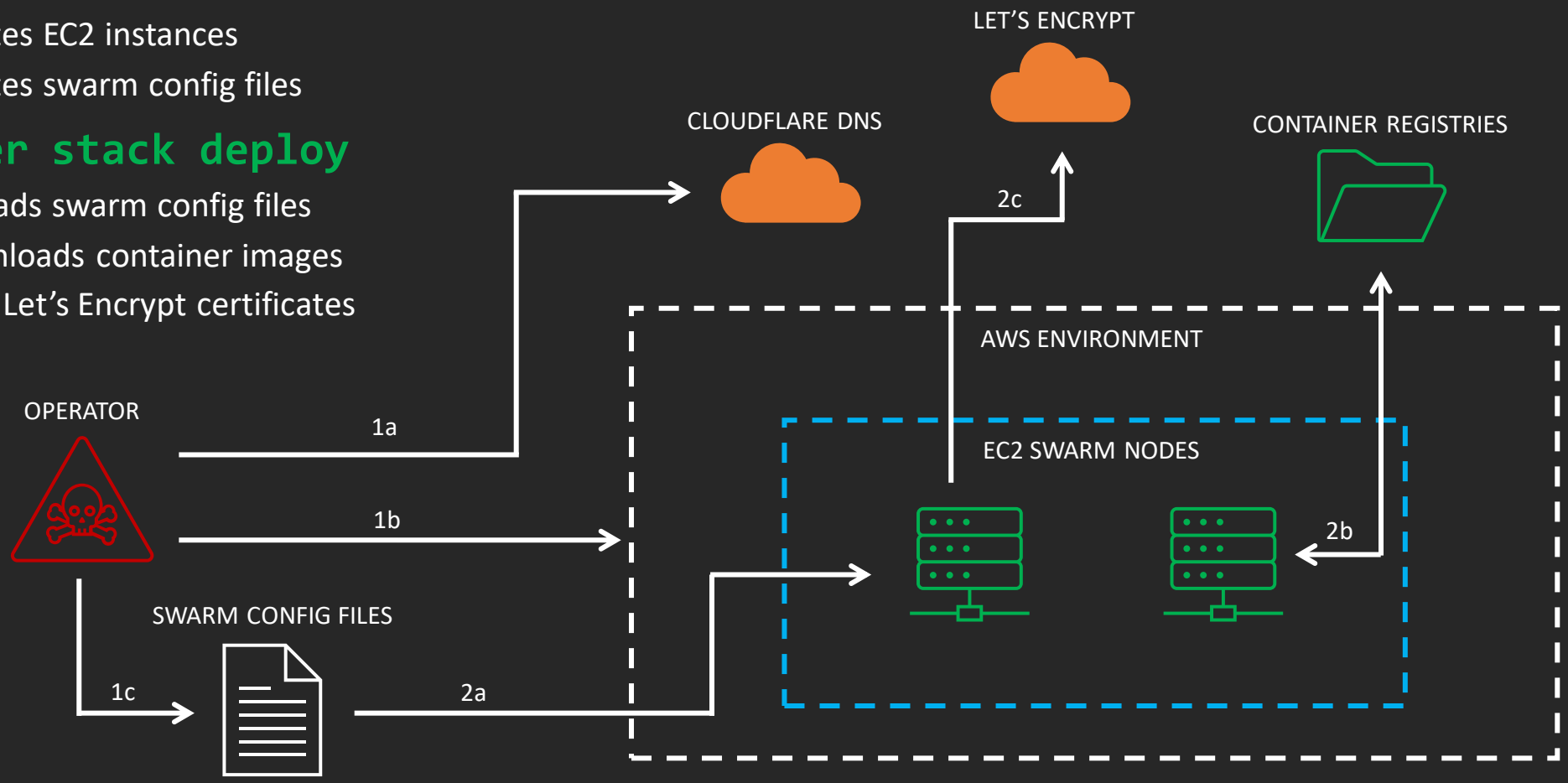
Deployment should be as turnkey as possible:

1. Set all the appropriate values in “terraform.tfvars”
2. `AWS_PROFILE=my_env terraform apply -auto-approve`
3. `docker -H ssh://ec2-user@10.10.10.10 stack deploy -c traefik.yaml Traefik`

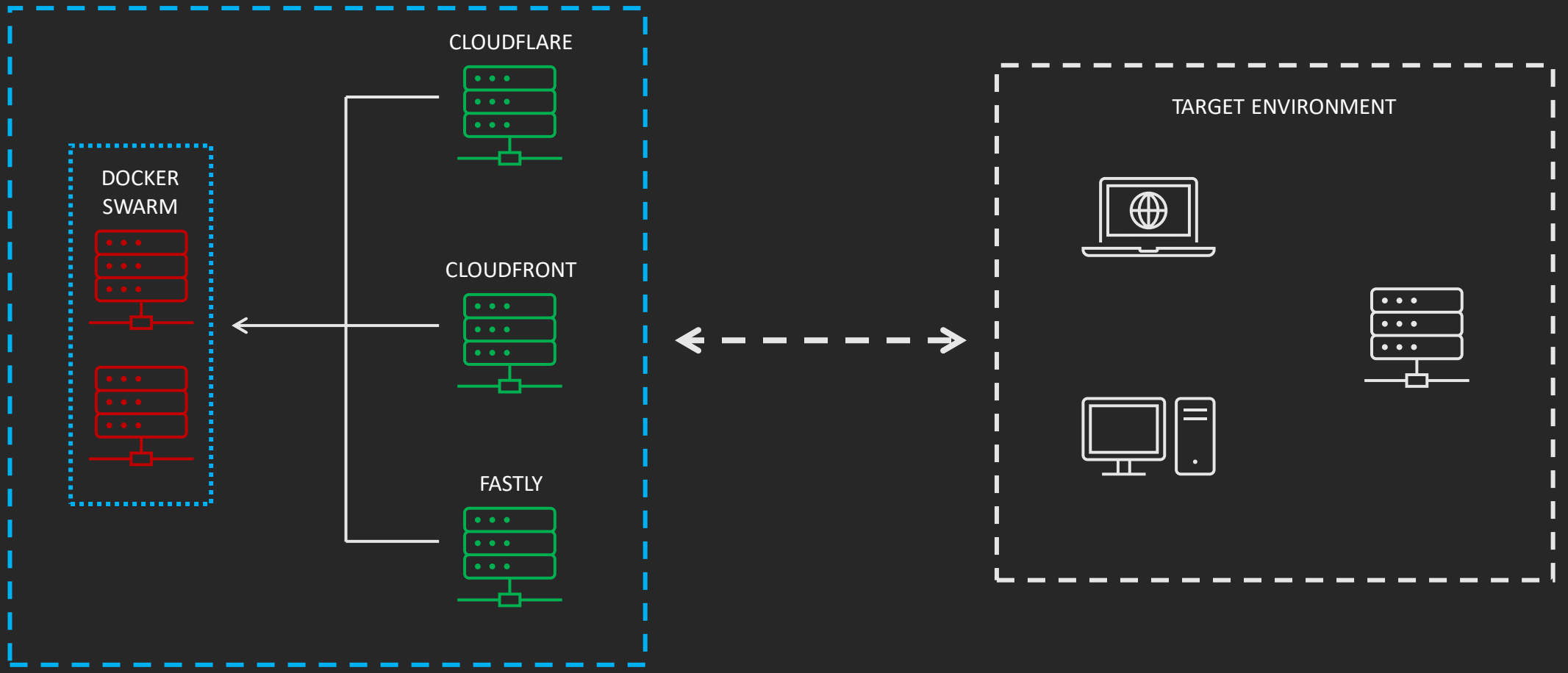
   Repeat the “stack deploy” for each desired service

Deployment Workflow

1. Run `terraform apply`
   a) Creates DNS records
   b) Creates EC2 instances
   c) Creates swarm config files
2. Run `docker stack deploy`
   a) Uploads swarm config files
   b) Downloads container images
   c) Mint Let's Encrypt certificates



Example Deployment



Ditch Ansible; Embrace Terraform

Arguable best-in-class IaC tooling

Provider agnostic

Minimal EC2 configuration is handled via userdata/cloudinit

- Docker Swarm initialization
- Mounting EFS
- Writing service configuration files to disk

```
## Generic Vars
project                = "GETINTHEBOX"
ingress_ip_cidrs       = [ "2.2.2.2/32", "3.3.3.0/24" ]
num_managers           = 1
num_workers            = 1
root_domain            = "legit.com"

## Traefik Vars
traefik_ui_domain      = "traefik.legit.com"

## Sliver Vars
sliver_image_name      = "somewhere/sliver:latest"
sliver_domains         = "`cdn.example.com`, `dist.example.com`"
sliver_profile_paths   = "`/api/v1/`, `/api/v2/`, `/api/v3/`"
sliver_profile_ua      = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0)
                              Gecko/20100101 Firefox/105.0"
```

```
## Generic Vars
project                = "GETINTHEBOX"
ingress_ip_cidrs       = [ "2.2.2.2/32", "3.3.3.0/24" ]
num_managers           = 1
num_workers            = 1
root_domain            = "legit.com"

## Traefik Vars
traefik_ui_domain      = "traefik.legit.com"

## Sliver Vars
sliver_image_name      = "somewhere/sliver:latest"
sliver_domains         = "`cdn.example.com`, `dist.example.com`"
sliver_profile_paths   = "`/api/v1/`, `/api/v2/`, `/api/v3/`"
sliver_profile_ua      = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0)
                               Gecko/20100101 Firefox/105.0"
```

```
## Generic Vars
project                = "GETINTHEBOX"
ingress_ip_cidrs       = [ "2.2.2.2/32", "3.3.3.0/24" ]
num_managers           = 1
num_workers            = 1
root_domain            = "legit.com"

## Traefik Vars
traefik_ui_domain      = "traefik.legit.com"

## Sliver Vars
sliver_image_name      = "somewhere/sliver:latest"
sliver_domains         = "`cdn.example.com`, `dist.example.com`"
sliver_profile_paths   = "`/api/v1/`, `/api/v2/`, `/api/v3/`"
sliver_profile_ua      = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0)
                              Gecko/20100101 Firefox/105.0"
```

```
## Generic Vars
project                 = "GETINTHEBOX"
ingress_ip_cidrs        = [ "2.2.2.2/32", "3.3.3.0/24" ]
num_managers            = 1
num_workers             = 1
root_domain             = "legit.com"

## Traefik Vars
traefik_ui_domain       = "traefik.legit.com"

## Sliver Vars
sliver_image_name       = "somewhere/sliver:latest"
sliver_domains          = "`cdn.example.com`, `dist.example.com`"
sliver_profile_paths    = "`/api/v1/`, `/api/v2/`, `/api/v3/`"
sliver_profile_ua       = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0)
                                Gecko/20100101 Firefox/105.0"
```

Docker Images

Core services (e.g., C2 server)

- Internally maintained
- Housed within private ECR instances
- Regular & automatic pipeline builds

Metaservices (e.g., Traefik)

- Maintained by “trusted” third-party
- Housed within Docker Hub
- Version pinned

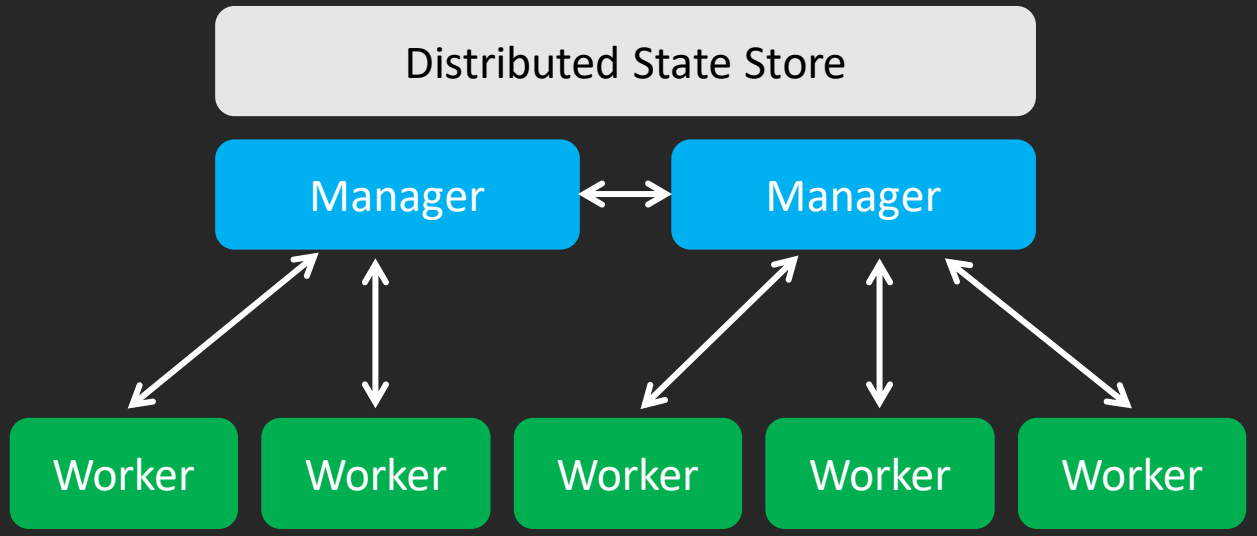
```
# Dockerfile
FROM golang:1.18.6
WORKDIR /opt
RUN git clone https://github.com/kgretzky/evilginx2.git && \
        cd evilginx2 && \
        make
COPY ./start.sh /opt
CMD [ "/opt/start.sh" ]


# start.sh
#!/bin/bash
screen -d -m /opt/evilginx2/bin/evilginx -p /opt/evilginx2/phishlets/
```

Docker Swarm

Container orchestration engine

Decouples applications from underlying servers

No, Kubernetes is not appropriate here!



Stacks

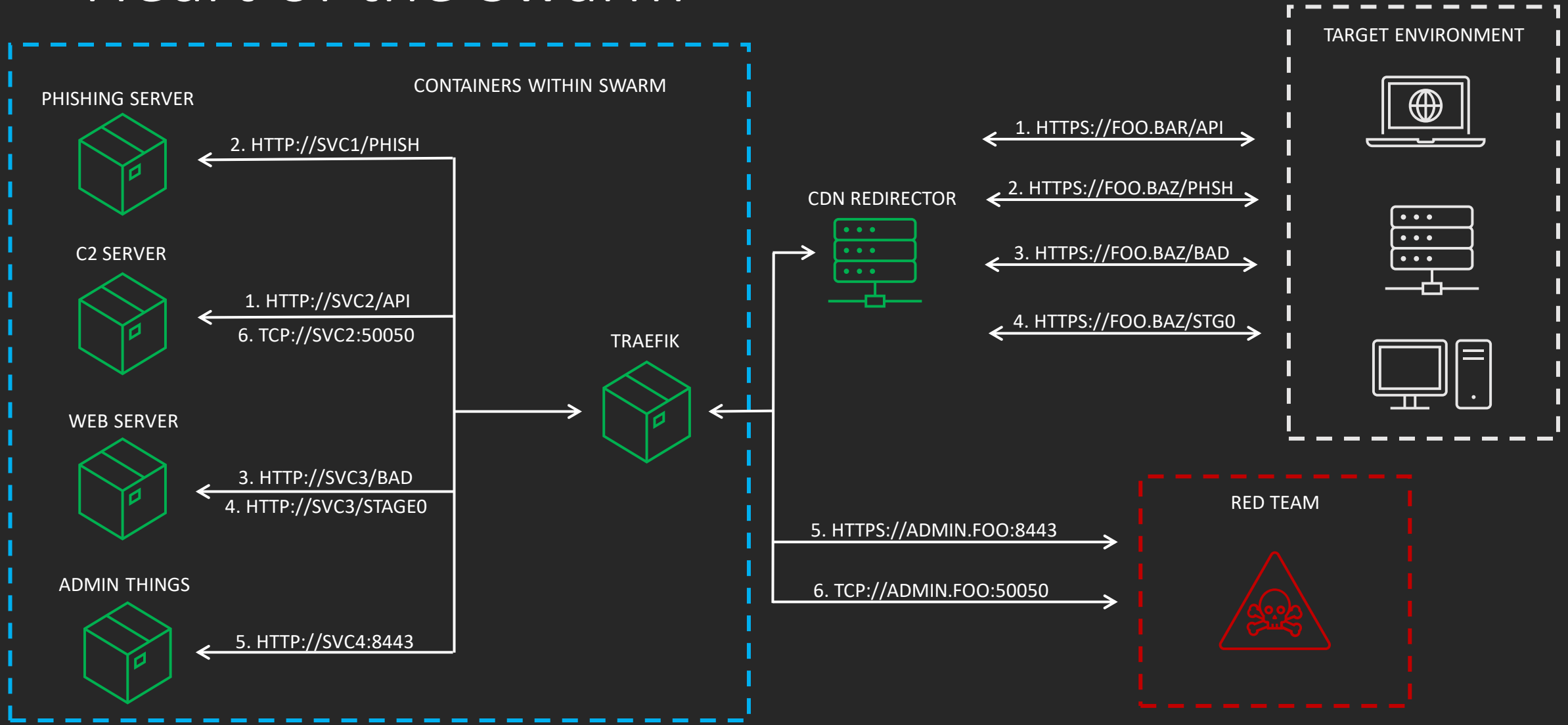
Groups images & network configurations together to act as a single service

Consumes Docker Compose files

Example deploy command:

```
docker -H ssh://ubuntu@swarm.host stack deploy –c
portainer.yaml portainer –with-registry-auth
```

Heart of the Swarm



Portainer

- Container used for easy management of the Swarm
- Helpful for deployment, management, debugging

Portainer Demo

Traefik and Routing


C2 traffic
Phishing traffic
Canary traffic
Traefik

Traefik

- Container used for HTTP/TCP/UDP routing
- Uses ports, rules, services, and routes to direct traffic
- Configure routes via container labels
- Auto-discover Swarm services
- Automatically mints & renews Let’s Encrypt certificates

```
labels:
  - traefik.enable=true
  - traefik.docker.network=public
  - traefik.constraint-label=public
  # primary profile
  - traefik.http.routers.sliver-profile.rule=Host(${sliver_profile_domains}) &&
      PathPrefix(${sliver_profile_paths}) && Headers(`User-Agent`,
      `${sliver_profile_ua}`)
  - traefik.http.routers.sliver-profile.entrypoints=https
  - traefik.http.routers.sliver-profile.tls=true
  - traefik.http.routers.sliver-profile.tls.certresolver=le
  - traefik.http.routers.sliver-profile.service=sliver-https
  - traefik.http.services.sliver-https.loadbalancer.server.port=443
  - traefik.http.services.sliver-https.loadbalancer.server.scheme=https
  # multiplayer connect
  - traefik.tcp.routers.sliver-multiplayer.rule=HostSNI(`*`)
  - traefik.tcp.routers.sliver-multiplayer.entrypoints=operator-ingress-tcp
  - traefik.tcp.routers.sliver-multiplayer=sliver-multiplayer
  - traefik.tcp.services.sliver-multiplayer.server.port=31337
```


```
labels:
  - traefik.enable=true
  - traefik.docker.network=public
  - traefik.constraint-label=public
  # primary profile
  - traefik.http.routers.sliver-profile.rule=Host(${sliver_profile_domains}) &&
      PathPrefix(${sliver_profile_paths}) && Headers(`User-Agent`,
      `${sliver_profile_ua}`)
  - traefik.http.routers.sliver-profile.entrypoints=https
  - traefik.http.routers.sliver-profile.tls=true
  - traefik.http.routers.sliver-profile.tls.certresolver=le
  - traefik.http.routers.sliver-profile.service=sliver-https
  - traefik.http.services.sliver-https.loadbalancer.server.port=443
  - traefik.http.services.sliver-https.loadbalancer.server.scheme=https
  # multiplayer connect
  - traefik.tcp.routers.sliver-multiplayer.rule=HostSNI(`*`)
  - traefik.tcp.routers.sliver-multiplayer.entrypoints=operator-ingress-tcp
  - traefik.tcp.routers.sliver-multiplayer=sliver-multiplayer
  - traefik.tcp.services.sliver-multiplayer.server.port=31337
```

```
labels:
  - traefik.enable=true
  - traefik.docker.network=public
  - traefik.constraint-label=public
  # primary profile
  - traefik.http.routers.sliver-profile.rule=Host(${sliver_profile_domains}) &&
      PathPrefix(${sliver_profile_paths}) && Headers(`User-Agent`,
      `${sliver_profile_ua}`)
  - traefik.http.routers.sliver-profile.entrypoints=https
  - traefik.http.routers.sliver-profile.tls=true
  - traefik.http.routers.sliver-profile.tls.certresolver=le
  - traefik.http.routers.sliver-profile.service=sliver-https
  - traefik.http.services.sliver-https.loadbalancer.server.port=443
  - traefik.http.services.sliver-https.loadbalancer.server.scheme=https
  # multiplayer connect
  - traefik.tcp.routers.sliver-multiplayer.rule=HostSNI(`*`)
  - traefik.tcp.routers.sliver-multiplayer.entrypoints=operator-ingress-tcp
  - traefik.tcp.routers.sliver-multiplayer=sliver-multiplayer
  - traefik.tcp.services.sliver-multiplayer.server.port=31337
```


```
labels:
  - traefik.enable=true
  - traefik.docker.network=public
  - traefik.constraint-label=public
  # primary profile
  - traefik.http.routers.sliver-profile.rule=Host(${sliver_profile_domains}) &&
      PathPrefix(${sliver_profile_paths}) && Headers(`User-Agent`,
      `${sliver_profile_ua}`)
  - traefik.http.routers.sliver-profile.entrypoints=https
  - traefik.http.routers.sliver-profile.tls=true
  - traefik.http.routers.sliver-profile.tls.certresolver=le
  - traefik.http.routers.sliver-profile.service=sliver-https
  - traefik.http.services.sliver-https.loadbalancer.server.port=443
  - traefik.http.services.sliver-https.loadbalancer.server.scheme=https
  # multiplayer connect
  - traefik.tcp.routers.sliver-multiplayer.rule=HostSNI(`*`)
  - traefik.tcp.routers.sliver-multiplayer.entrypoints=operator-ingress-tcp
  - traefik.tcp.routers.sliver-multiplayer=sliver-multiplayer
  - traefik.tcp.services.sliver-multiplayer.server.port=31337
```

```
labels:
  - traefik.enable=true
  - traefik.docker.network=public
  - traefik.constraint-label=public
  # primary profile
  - traefik.http.routers.sliver-profile.rule=Host(${sliver_profile_domains}) &&
      PathPrefix(${sliver_profile_paths}) && Headers(`User-Agent`,
      `${sliver_profile_ua}`)
  - traefik.http.routers.sliver-profile.entrypoints=https
  - traefik.http.routers.sliver-profile.tls=true
  - traefik.http.routers.sliver-profile.tls.certresolver=le
  - traefik.http.routers.sliver-profile.service=sliver-https
  - traefik.http.services.sliver-https.loadbalancer.server.port=443
  - traefik.http.services.sliver-https.loadbalancer.server.scheme=https
  # multiplayer connect
  - traefik.tcp.routers.sliver-multiplayer.rule=HostSNI(`*`)
  - traefik.tcp.routers.sliver-multiplayer.entrypoints=operator-ingress-tcp
  - traefik.tcp.routers.sliver-multiplayer=sliver-multiplayer
  - traefik.tcp.services.sliver-multiplayer.server.port=31337
```

4

Monitoring

Infrastructure Monitoring

Infrastructure telemetry

- Traefik access logs
- System and service health

Application telemetry

- Phishing logs
- Operator logs
- Canary logs

Logging & Metrics

Prometheus + Loki

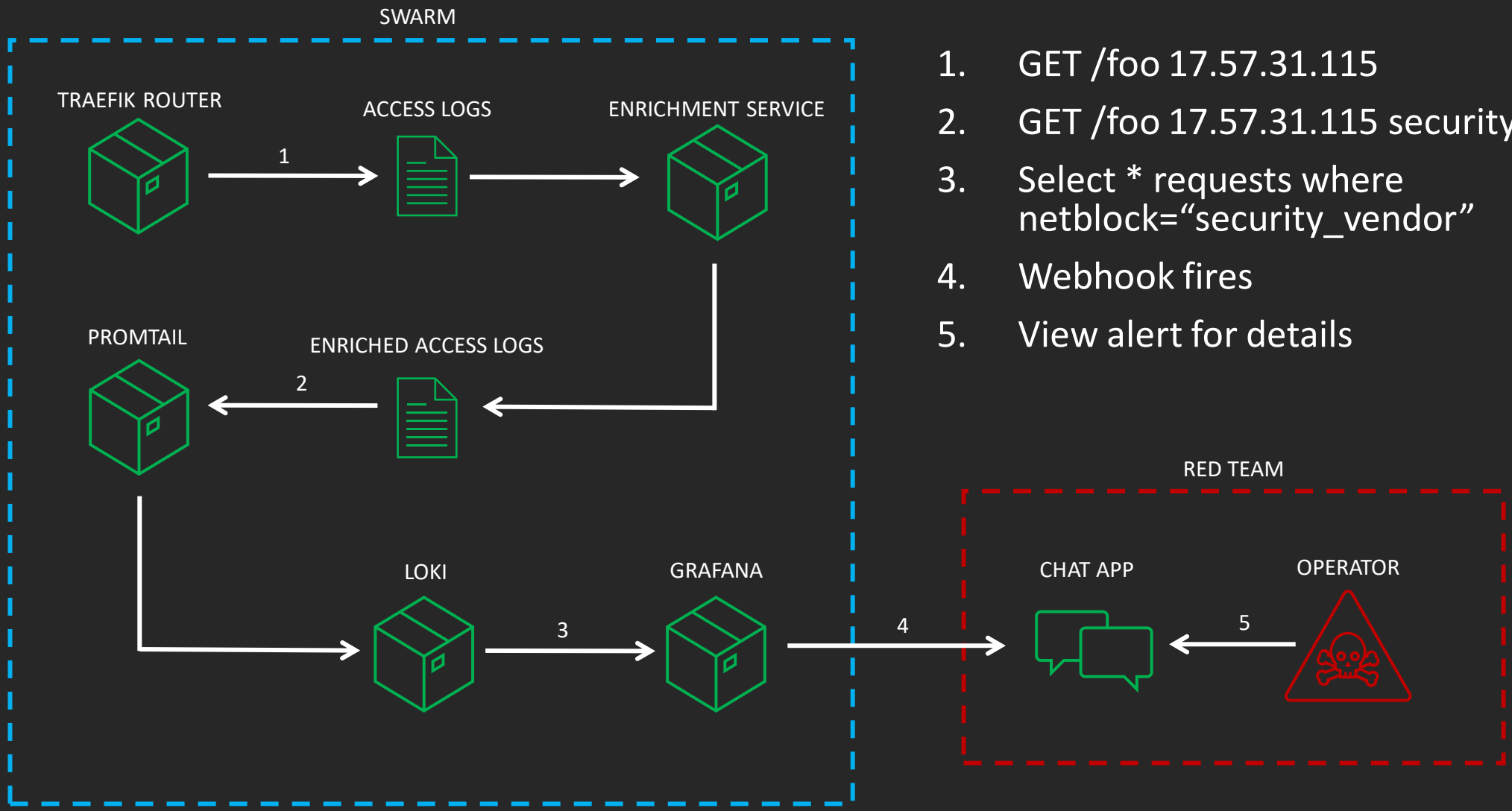
- ✓ Lightweight
- ✓ Easily integrates with our stack
- ✓ Infrastructure setup is turnkey
- ✓ Log ingestion is turnkey
- ✗ Limited pipeline flexibility

Red Team SIEM || RedELK

- ✗ ELK is resource intensive
- ✗ Elasticsearch is complex
- ✓ Infrastructure setup is turnkey
- ✗ Log ingestion is not turnkey
- ✓ Log parsing for most red tools

https://github.com/outflanknl/RedELK
https://github.com/SecurityRiskAdvisors/RedTeamSIEM

Monitoring Workflow



1. GET /foo 17.57.31.115
2. GET /foo 17.57.31.115 security_vendor
3. Select * requests where netblock=“security_vendor”
4. Webhook fires
5. View alert for details

Closing Thoughts

Lessons Learned

Traefik is best suited for HTTP

- UDP/TCP routing has significant limitations
- Overlay networks can complicate true source IP preservation

EC2 User Data is a chore to debug

- cat /var/log/cloud-init-output.log

Infrastructure Thoughts

Just because someone did it one way doesn’t mean you have to

Look for ways to simplify and reduce stress in your life (with infra)

Explore new technologies and tools

Be a hacker :)

Resources

https://github.com/SecurityRiskAdvisors/GetInTheBox

https://grafana.com/docs/loki/latest/getting-started/

https://doc.traefik.io/traefik/

https://docs.docker.com/engine/swarm/


Questions?