aws

COMMUNITY DAY

NORDICS

Trustpilot

CYBERCOM GROUP

QUEUE·IT

TRADESHIFT

VEEAM N2WS

Efficiently exposing apps on Kubernetes at scale

Rasheed Amir, Stakater

aws
COMMUNITY DAY
NORDICS

Problem

Kubernetes runs container workloads in Pods

… but these are not automatically accessible outside the cluster

- ❖ What options does Kubernetes provide for this?
- ❖ How do we utilize these options efficiently?
  - ➢ across multiple apps (e.g. for micro-frontends)
  - ➢ across redeployments (e.g. for continuous deployment)

Stakater

Agenda

We will explore...

The basics of how to expose an app on Kubernetes

How to use automation to scale the process for multiple apps

Some useful best practices for these tools and processes

Stakater

About me

kubernetes + OPENSHIFT = Stakater

Stakater

About Stakater

Based in **Stockholm**

https://github.com/stakater

Kubernetes Expert! Team of professionals experienced with DevOps Automation and Full-stack web application development

We provide professional tools and services to help customers create and manage their Kubernetes based infrastructure effortlessly

Some of our clients: ATG ERICSSON Försäkringskassan SCANIA

Service

What is a Kubernetes Service?

- An abstraction which provides access to a logical set of Pods
- Pods come and go, but Service has a stable IP address
- Provides load balancing (primitive) across member pods
- Which pods?
  - Determined by label selector
- How to access?
  - Determined by service type

Service Type: ClusterIP

Stakater

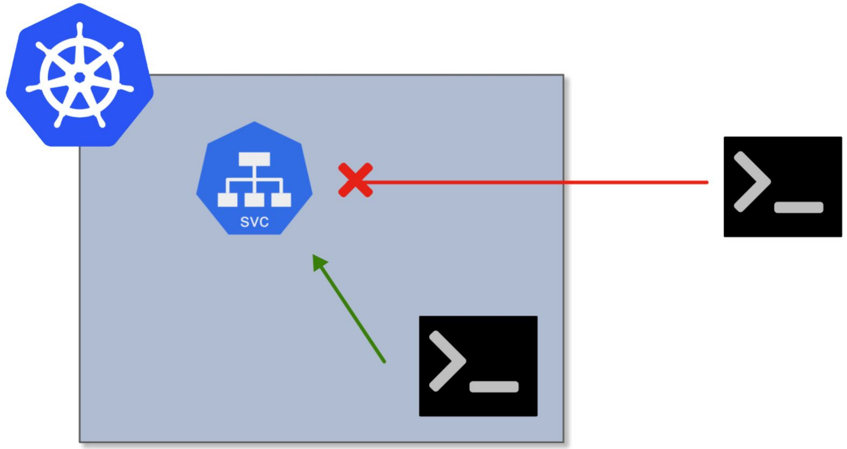
ClusterIP

- Default service type
- Service is accessible on a cluster internal IP
- Apps inside the cluster can access the service

ClusterIP

But...

❖ No access from outside the cluster



Service Type: NodePort

Stakater

NodePort

❖ exposes the service on a static port on each node

NodePort

But...

- ❖ can only have one service per port
- ❖ a limited number of usable ports
- ❖ Needs special handling for cases of change in Node/VM IP

Stakater

Service Type: LoadBalancer

Stakater

LoadBalancer

- ❖ exposes the app using a cloud provider’s network load balancer
- ❖ The load balancer gets a single IP

LoadBalancer

But...

- all traffic on the port will be forwarded to the service. no filtering or routing.
- each service exposed is handled by a separate Load Balancer.
  - **Skyrocketing cost in a large scale application.**

Stakater

Ingress

Ingress

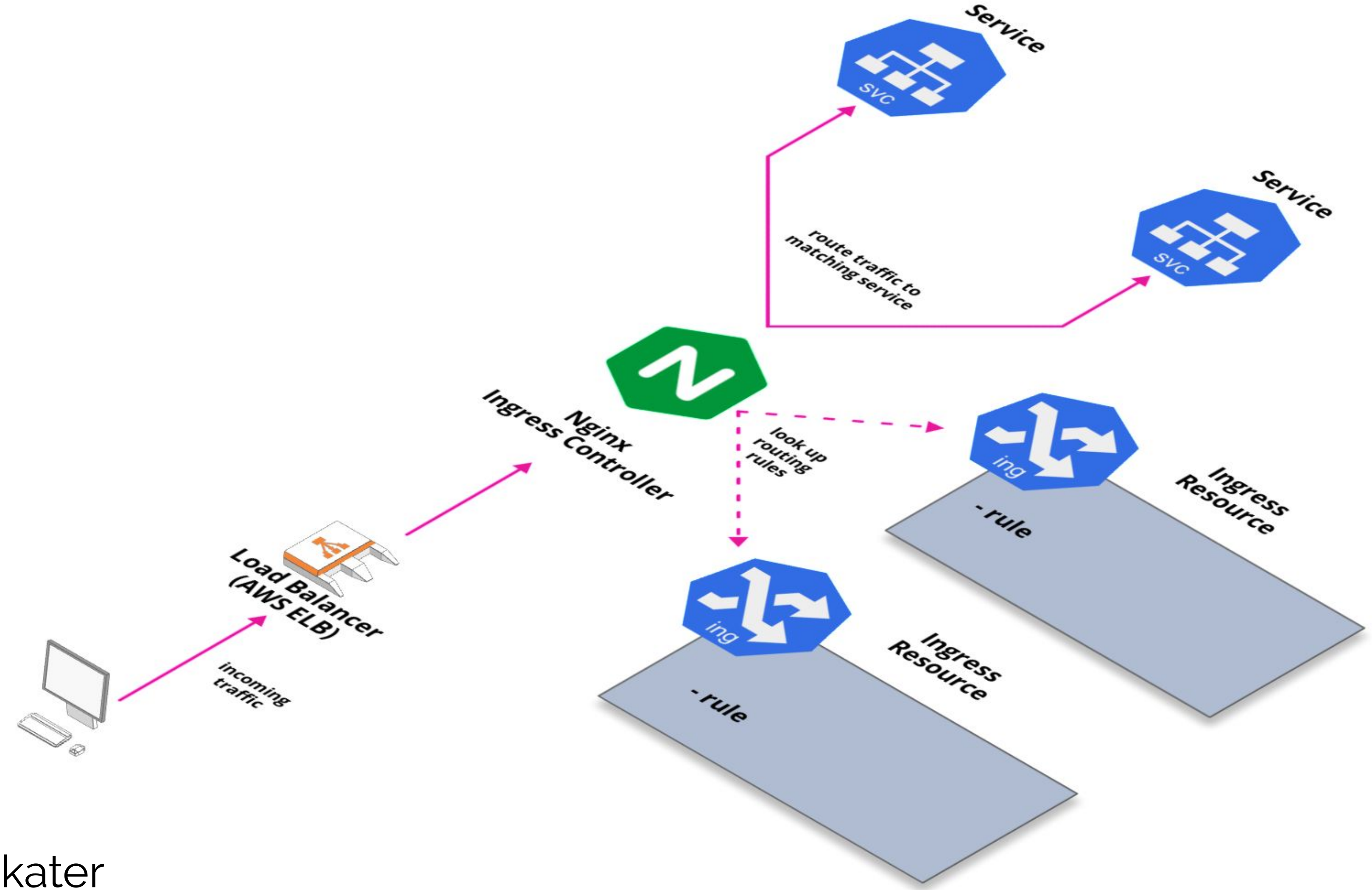
- More efficient way of exposing services
- Route traffic based on the request host or path
- Centralization of many services to a single point
- Use ClusterIP Service type

Ingress Controller

- ❖ Required by Ingress to work
- ❖ looks up Ingress resource definitions and routes traffic to services accordingly
- ❖ match with Ingress based on Class name

nginx ingress controller

- ❖ automatically creates a Load Balancer, e.g. ELB for AWS
- ❖ SSL termination
- ❖ Load balancing


Service
Service
route traffic to matching service
svc
svc
Nginx Ingress Controller
look up routing rules
ing
ing
Ingress Resource
Ingress Resource
- rule
- rule
Load Balancer (AWS ELB)
incoming traffic

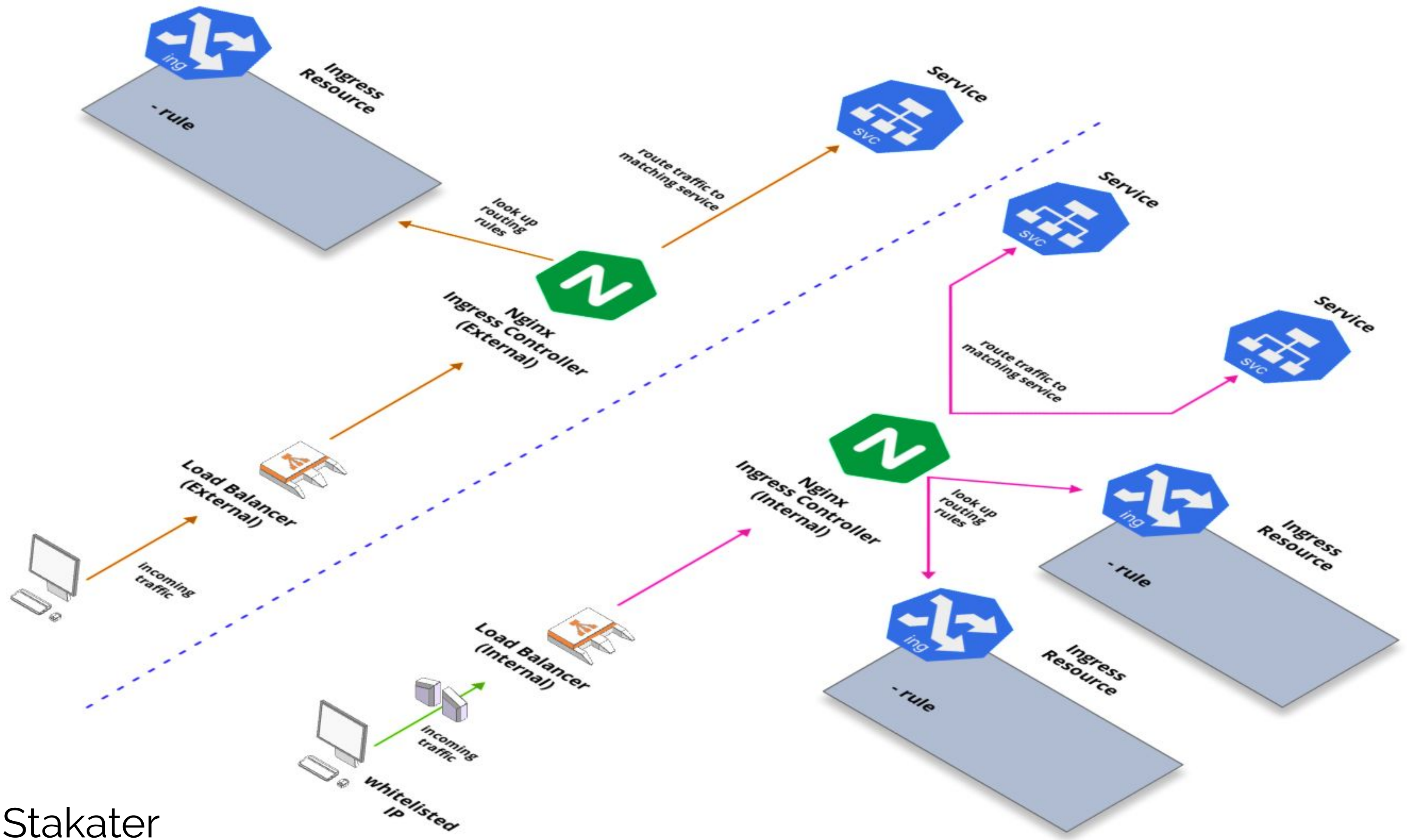
Stakater

Best practice

Ingress Controller

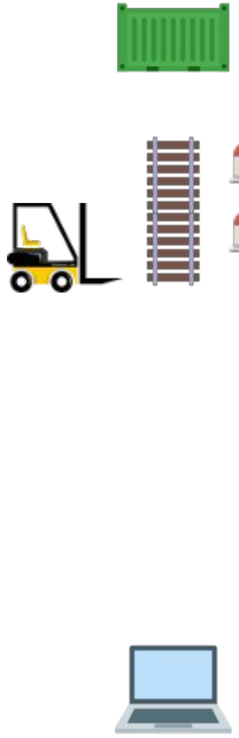
- 2 ingress controllers and 2 load balancers
  - one for public applications
  - second for private applications
- private applications and load balancer should have restricted access
  - security groups, IP whitelisting, etc.

Stakater


Ingress Resource
- rule
Service
route traffic to matching service
look up routing rules
Nginx Ingress Controller (External)
Service
Service
route traffic to matching service
Load Balancer (External)
Nginx Ingress Controller (Internal)
look up routing rules
Ingress Resource
- rule
incoming traffic
Load Balancer (Internal)
Ingress Resource
- rule
incoming traffic
whitelisted IP

Stakater

Checkpoint



Stakater

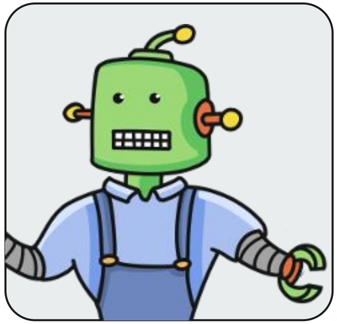
Let's Reflect

Manually creating ingress resource for each application...

...is too much manual work

How do we do it efficiently for all applications?

Let's Automate!



Stakater

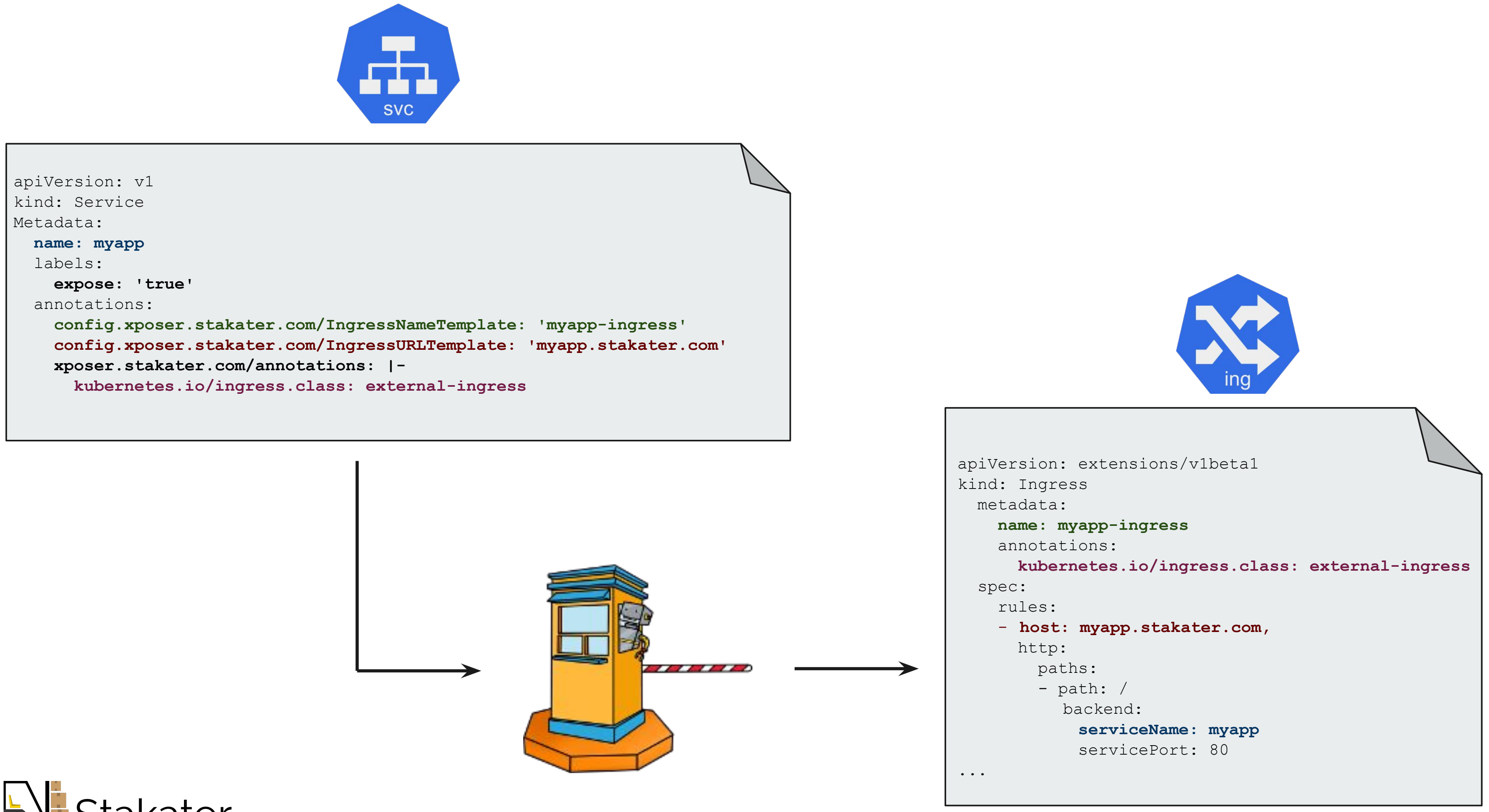
Stakater Xposer

https://github.com/stakater/Xposer

Stakater

Stakater Xposer

- ❖ Automatically creates/updates/deletes an ingress for a service with config from annotations
- ❖ Optionally uses CertManager to automatically generate TLS certificates



```yaml
apiVersion: v1
kind: Service
Metadata:
  name: myapp
  labels:
    expose: 'true'
  annotations:
    config.xposer.stakater.com/IngressNameTemplate: 'myapp-ingress'
    config.xposer.stakater.com/IngressURLTemplate: 'myapp.stakater.com'
    xposer.stakater.com/annotations: |-
      kubernetes.io/ingress.class: external-ingress
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
  metadata:
    name: myapp-ingress
    annotations:
      kubernetes.io/ingress.class: external-ingress
  spec:
    rules:
    - host: myapp.stakater.com,
      http:
        paths:
        - path: /
          backend:
            serviceName: myapp
            servicePort: 80
...
```

Stakater

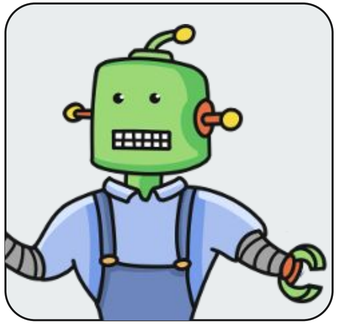
Next step

The load balancer will have an auto-generated unfriendly domain name.

```
b8d03a52e6b8611e98c4d02a061b92d1-1200162703.us-west-2.elb.amazonaws.com
```

We would like to use our custom domain name.

What do we do?



DNS!

Domain Name Systems (DNS)

What is DNS

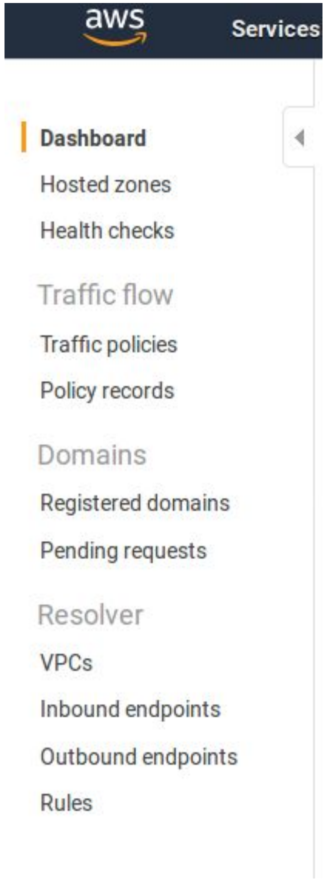
- The phonebook of the Internet
- translates domain names e.g. aws.amazon.com to IP addresses so browsers can load Internet resources
- DNS Servers hold these records

AWS Route53

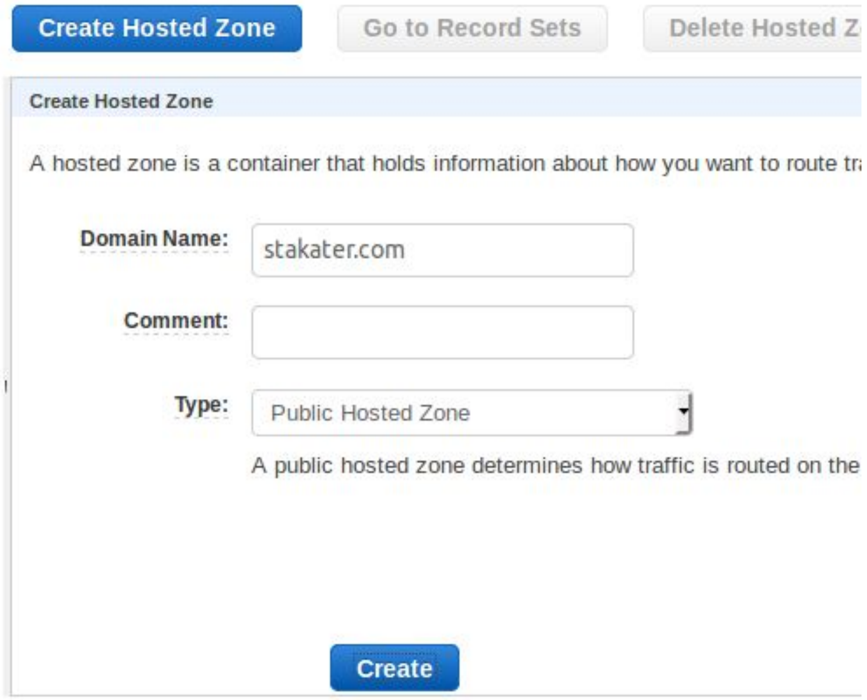
Stakater

What is Route53

- ❖ Amazon's Domain Name System (DNS) web service
- ❖ Main functions
  - ➢ domain registration
  - ➢ **DNS routing**
  - ➢ health checking



Create Hosted Zone



Create Hosted Zone

A hosted zone is a container that holds information about how you want to route tr

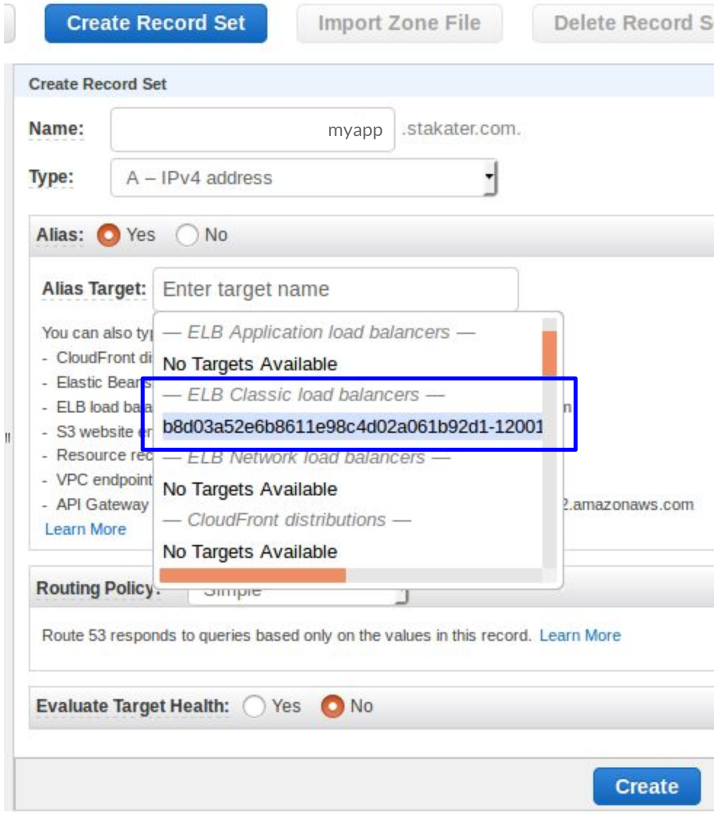
Domain Name: stakater.com

Comment:

Type: Public Hosted Zone

A public hosted zone determines how traffic is routed on the

Create Record Set



Create Record Set

Name: myapp .stakater.com.

Type: A – IPv4 address

Alias: [✓ Yes] No

Alias Target: Enter target name

You can also typ
- CloudFront di
- Elastic Beans
- ELB load bala
- S3 website e
- Resource rec
- VPC endpoint
- API Gateway

Learn More

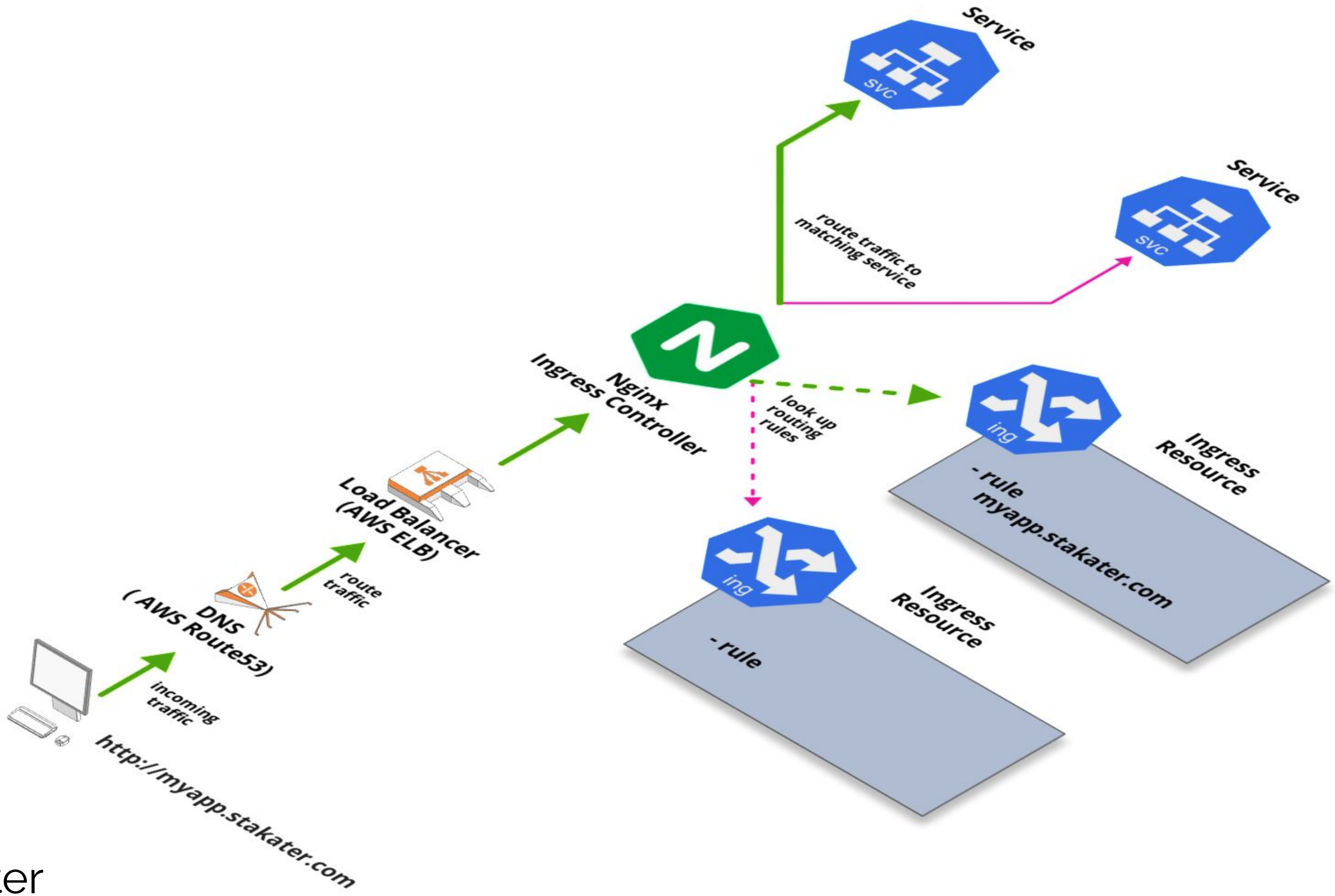
— ELB Application load balancers —
No Targets Available
— ELB Classic load balancers —
b8d03a52e6b8611e98c4d02a061b92d1-12001
— ELB Network load balancers —
No Targets Available
— CloudFront distributions —
No Targets Available

2.amazonaws.com

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. Learn More

Evaluate Target Health: Yes [✓ No]


Service
Service
route traffic to matching service
Nginx Ingress Controller
look up routing rules
Ingress Resource
- rule myapp.stakater.com
Ingress Resource
- rule
Load Balancer (AWS ELB)
route traffic
DNS (AWS Route53)
incoming traffic
http://myapp.stakater.com

Stakater

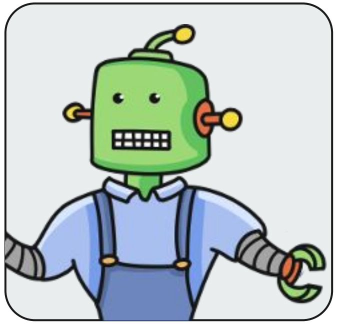
Let's Reflect

Manually creating DNS records for each service...

...is too much manual work

How do we do it efficiently for all applications?

Let's Automate!



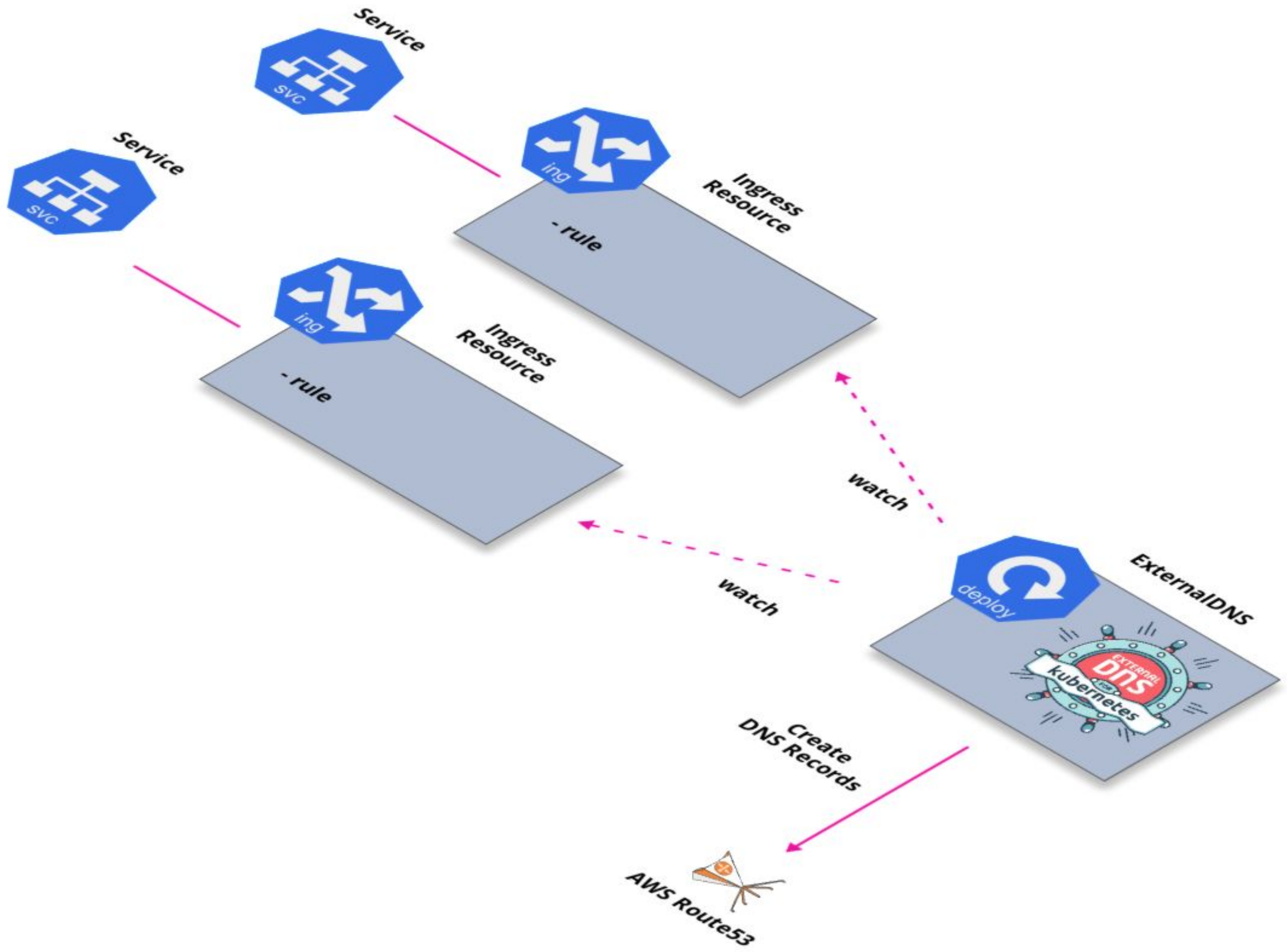
Stakater

ExternalDNS

https://github.com/kubernetes-incubator/external-dns

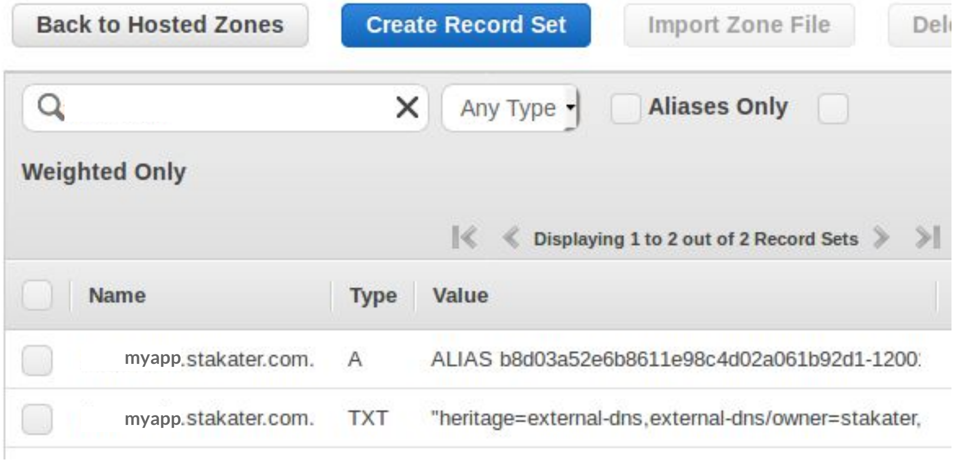
ExternalDNS

- Automates DNS entries for our application deployments
- Configures DNS records by looking at the resources (Services, Ingresses, etc.)
- Keeps DNS entries in sync
  - add DNS entries for a new exposed app
  - clean up entries when the app is removed from the cluster.


Service
svc
Service
svc
ing
Ingress Resource
- rule
ing
Ingress Resource
- rule
watch
watch
deploy
ExternalDNS
EXTERNAL DNS for kubernetes
Create DNS Records
AWS Route53

Stakater

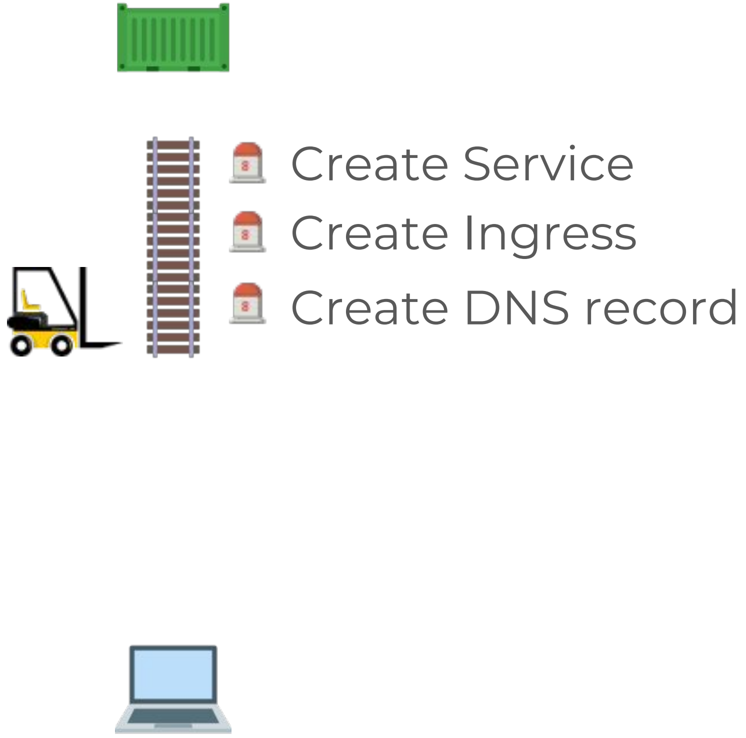
```
apiVersion: extensions/v1beta1,
kind: Ingress,
  metadata: {
        name: myapp-ingress,
  }
...
    rules: [
      {
        host: myapp.stakater.com,
        http: {
          paths: [
            {
...
```



Name	Type	Value
myapp.stakater.com.	A	ALIAS b8d03a52e6b8611e98c4d02a061b92d1-1200:
myapp.stakater.com.	TXT	"heritage=external-dns,external-dns/owner=stakater,

Stakater

Checkpoint



- Create Service
- Create Ingress
- Create DNS record

Stakater

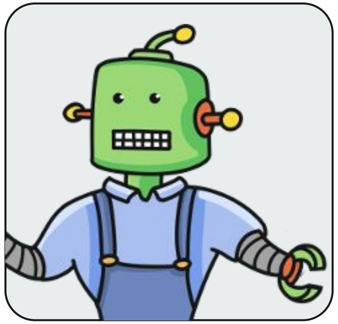
Next step

The connection to our service is not secure

We are accessing it over http and not https

We would like our service to be accessed over a secure connection.

What do we do?



TLS!

Stakater

TLS Certificates

What is TLS (Transport Layer Security)

- Previously called SSL
- security protocol for communications over the Internet
- HTTPS is TLS encryption on top of HTTP
- primary use case is securing communication between web clients and servers
  - TLS Certificate
    - facilitates the encrypted connection
    - Used for validating the website identity
    - Issued from a Certificate Authority

Cert Manager

https://github.com/jetstack/cert-manager

Stakater

Cert Manager

- ❖ automate the management and issuance of TLS certificates
- ❖ attempt to renew certificates at an appropriate time before expiry
- ❖ Certificate issuers at namespace or cluster-wide level
- ❖ Free Certificate Issuers e.g. Let's Encrypt
- ❖ Certificate installed on Ingress

Stakater

Cert Manager

However...

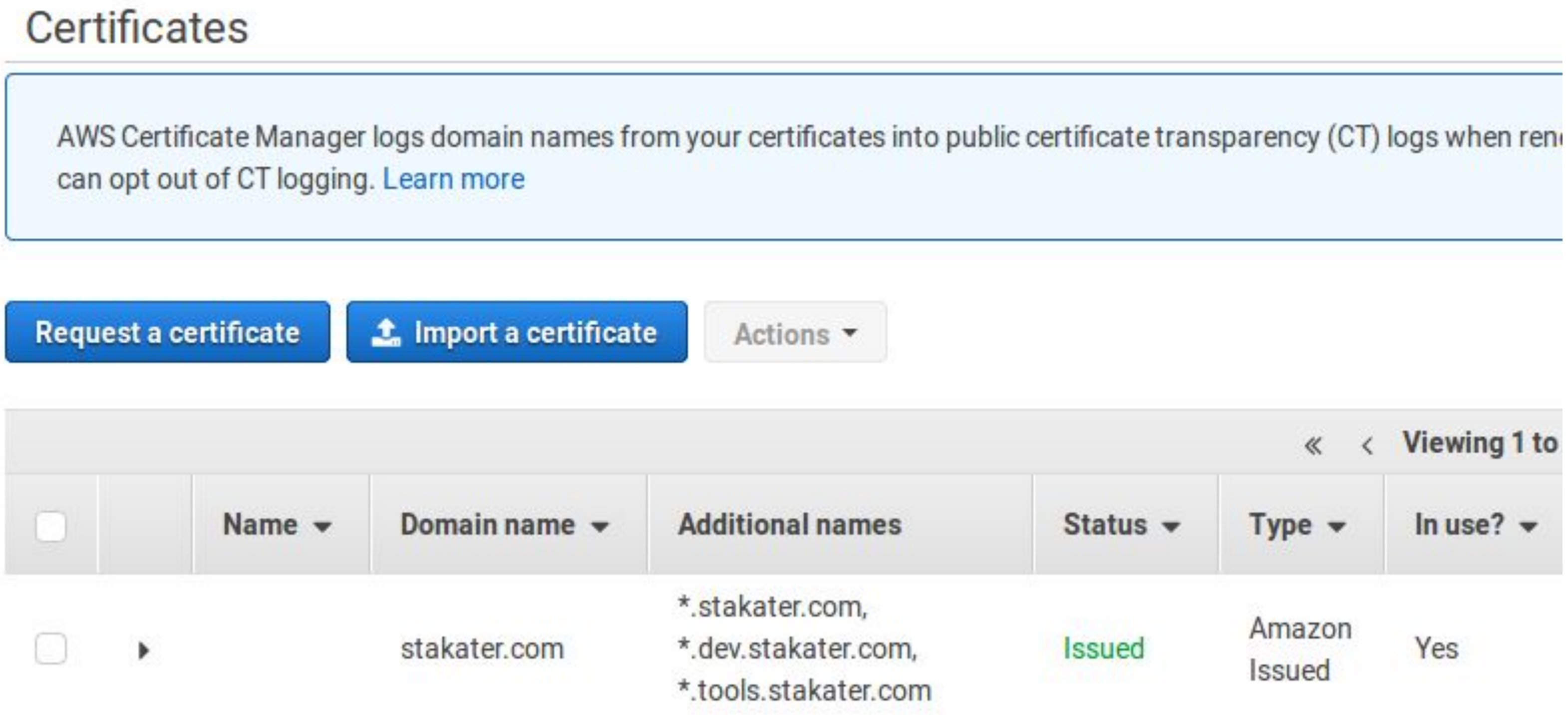
- Free Certificate issuers may have restrictions
  - Let's Encrypt
    - 50 Certificates per Registered Domain per week
    - 5 Duplicate Certificates per week
  - Redeploying Ingresses will require Certificate re-issue

AWS Certificate Manager (ACM)

Stakater

AWS Certificate Manager (ACM)

- Easily provision, manage, and deploy SSL/TLS certificates
  - Quickly request certificate
  - Quickly deploy it on AWS resources e.g. ELB
- AWS Certificate Manager handles certificate renewals
- Installed on the Load Balancer; reissuing won't be that often



Certificates

AWS Certificate Manager logs domain names from your certificates into public certificate transparency (CT) logs when ren
can opt out of CT logging. Learn more

		Name	Domain name	Additional names	Status	Type	In use?
			stakater.com	*.stakater.com, *.dev.stakater.com, *.tools.stakater.com	Issued	Amazon Issued	Yes

Stakater

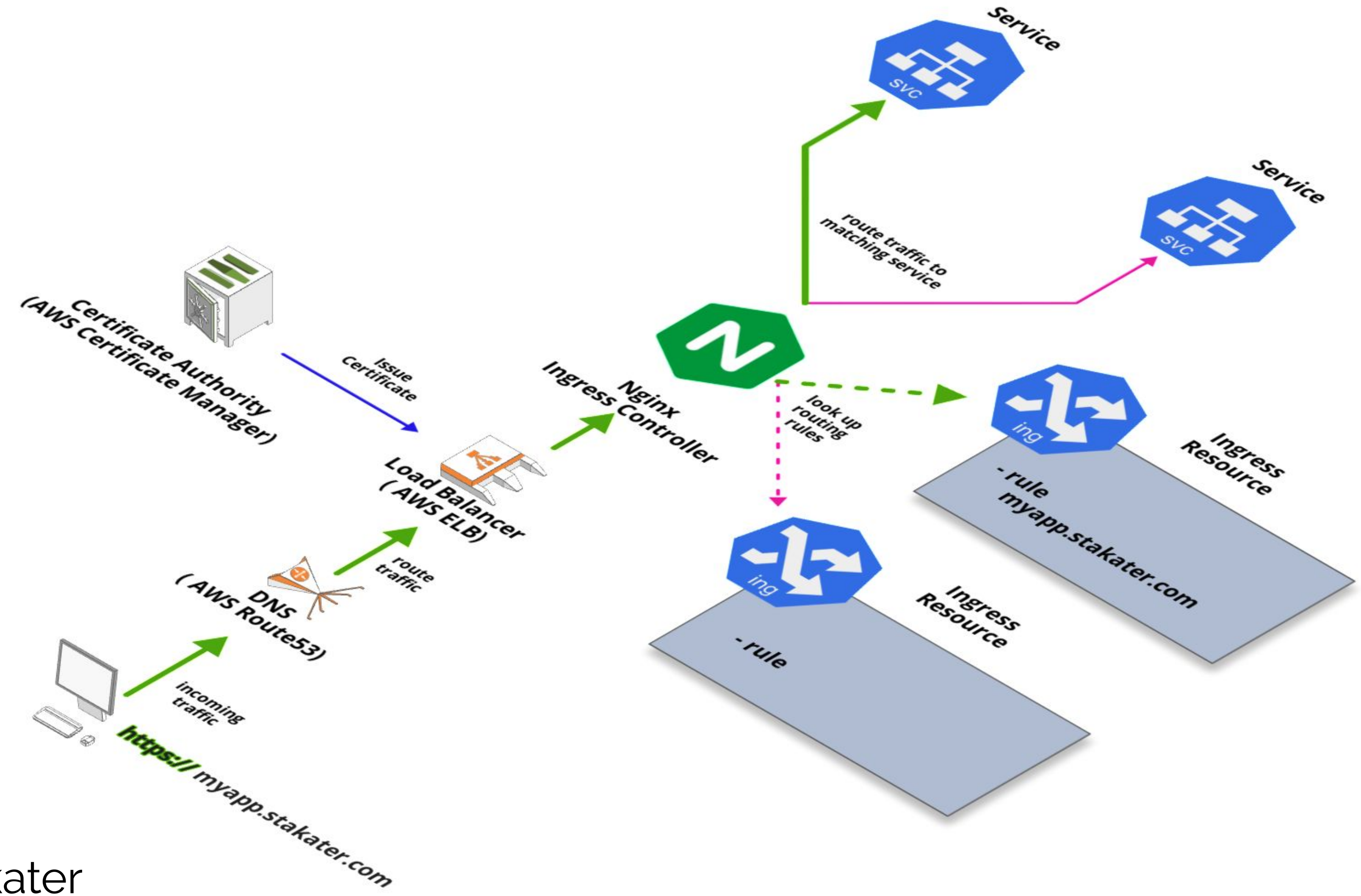
Best practice

AWS Certificate Manager (ACM)

Automate issuing or re-issuing certificates

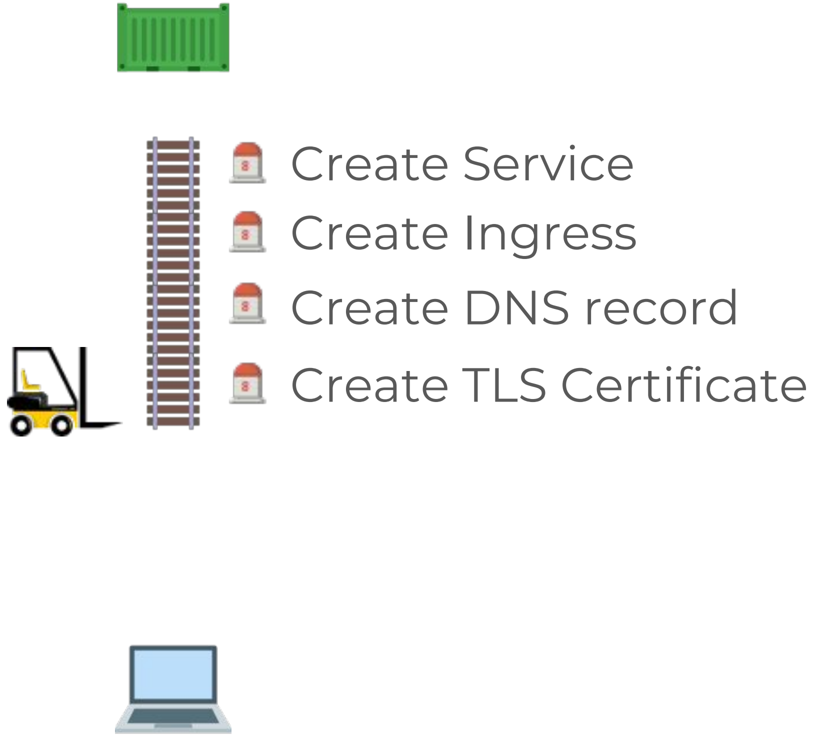
- ❖ Terraform
- ❖ AWS Service Operator
  - ➢ Recently developed
  - ➢ ACM not yet supported, but planned
  - ➢ Preferable to use once ACM is integrated

Stakater


Service
Service
route traffic to matching service
Certificate Authority (AWS Certificate Manager)
Issue Certificate
Nginx Ingress Controller
look up routing rules
Ingress Resource
- rule myapp.stakater.com
Load Balancer (AWS ELB)
route traffic
DNS (AWS Route53)
Ingress Resource
- rule
incoming traffic
https:// myapp.stakater.com
svc
svc
ing
ing

Stakater

Checkpoint



Stakater

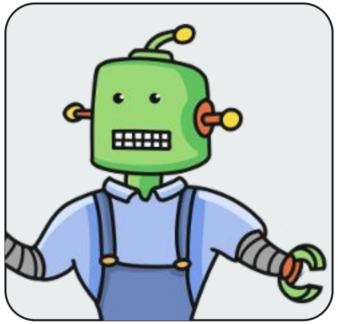
Next step

Our service is now securely accessible

How do we ensure its uptime?

and get notified if it goes down?

Monitoring!



Stakater

Uptime Monitoring

Uptime Monitoring

- Continually check reachability of app from global locations
- Uptime Checkers
  - UptimeRobot
    - 50 free monitors
  - Pingdom
  - Statuscake
  - Others...

Best practice

Uptime Monitoring

- Verify from multiple locations across the globe
- Frequent checks for production services
- Infrequent checks for non-production services
- Use instant alerts, e.g. Slack, etc.

Stakater

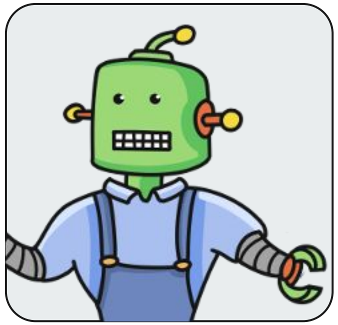
Let's Reflect

Manually creating Uptime monitors for each service...

...is much manual work

How do we do it efficiently for all applications?

Let's Automate!



Stakater

Stakater Ingress Monitor Controller (IMC)

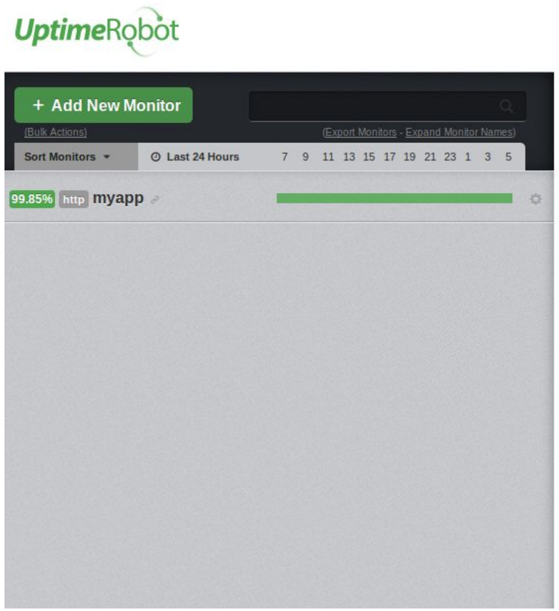
https://github.com/stakater/IngressMonitorController

Stakater

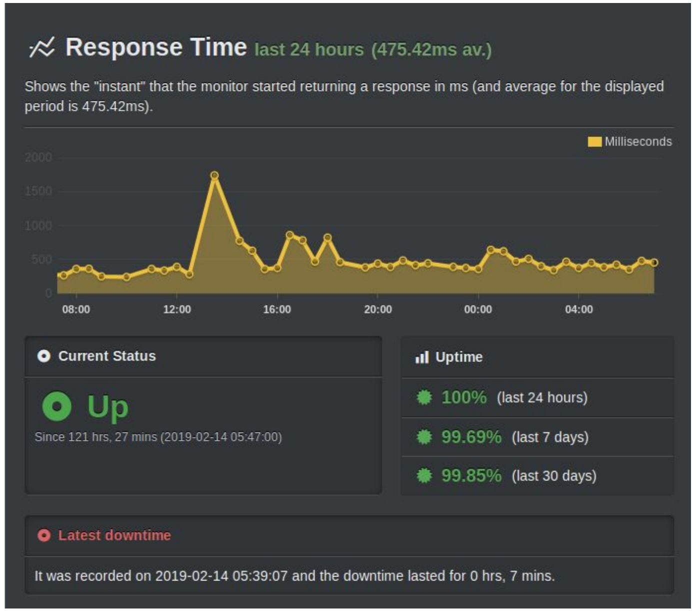
What is IMC

- automatically add / remove monitors against ingresses in the uptime checker
- Uptime checker monitors the endpoint and alert when down
- Notification channels configured in Uptime checker
  - Slack
  - Email

```
apiVersion: extensions/v1beta1,
kind: Ingress,
metadata:
  name: myapp
  annotations:
    monitor.stakater.com/enabled: true
...
```



99.85% http myapp



Response Time last 24 hours (475.42ms av.)

Shows the "instant" that the monitor started returning a response in ms (and average for the displayed period is 475.42ms).

Current Status

Up

Since 121 hrs, 27 mins (2019-02-14 05:47:00)

Uptime

100% (last 24 hours)

99.69% (last 7 days)

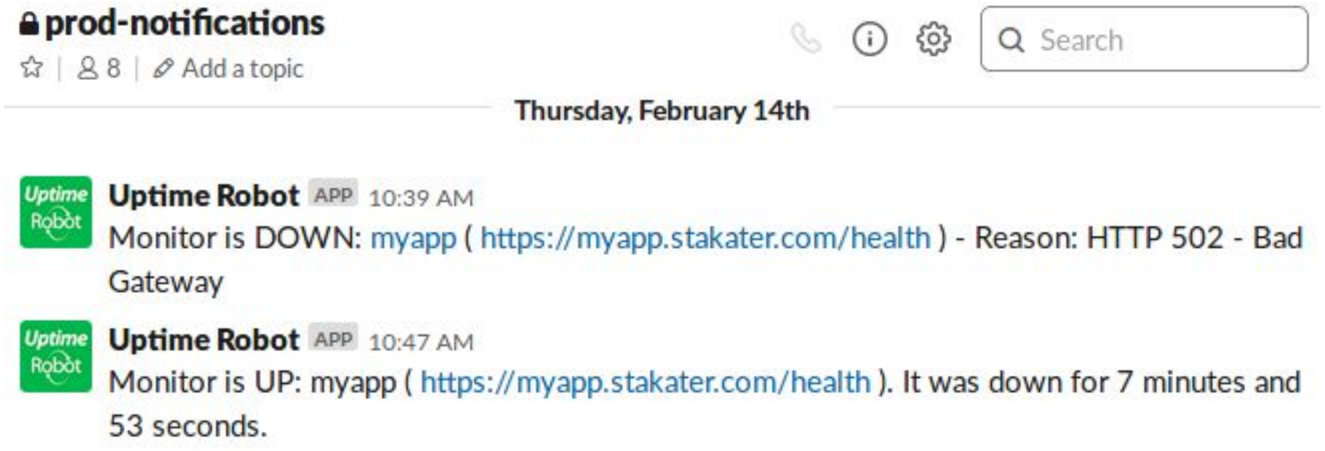
99.85% (last 30 days)

Latest downtime

It was recorded on 2019-02-14 05:39:07 and the downtime lasted for 0 hrs, 7 mins.

Stakater

Slack alerts

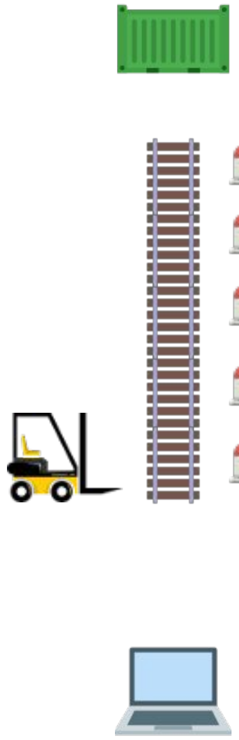


Thursday, February 14th

Uptime Robot APP 10:39 AM
Monitor is DOWN: myapp (https://myapp.stakater.com/health) - Reason: HTTP 502 - Bad Gateway

Uptime Robot APP 10:47 AM
Monitor is UP: myapp (https://myapp.stakater.com/health). It was down for 7 minutes and 53 seconds.

Checkpoint



- Create Service
- Create Ingress
- Create DNS record
- Create TLS Certificate
- Create Uptime Monitor

Stakater

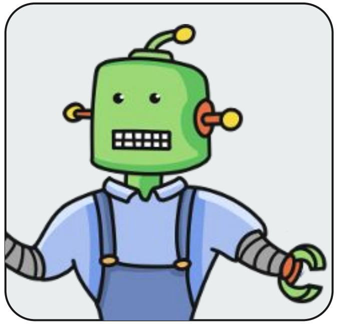
Let's Reflect

Keeping track of multiple services and where to access them...

...can be difficult

How do we efficiently keep track of all applications?

Let's Automate!



Stakater

Stakater Forecastle

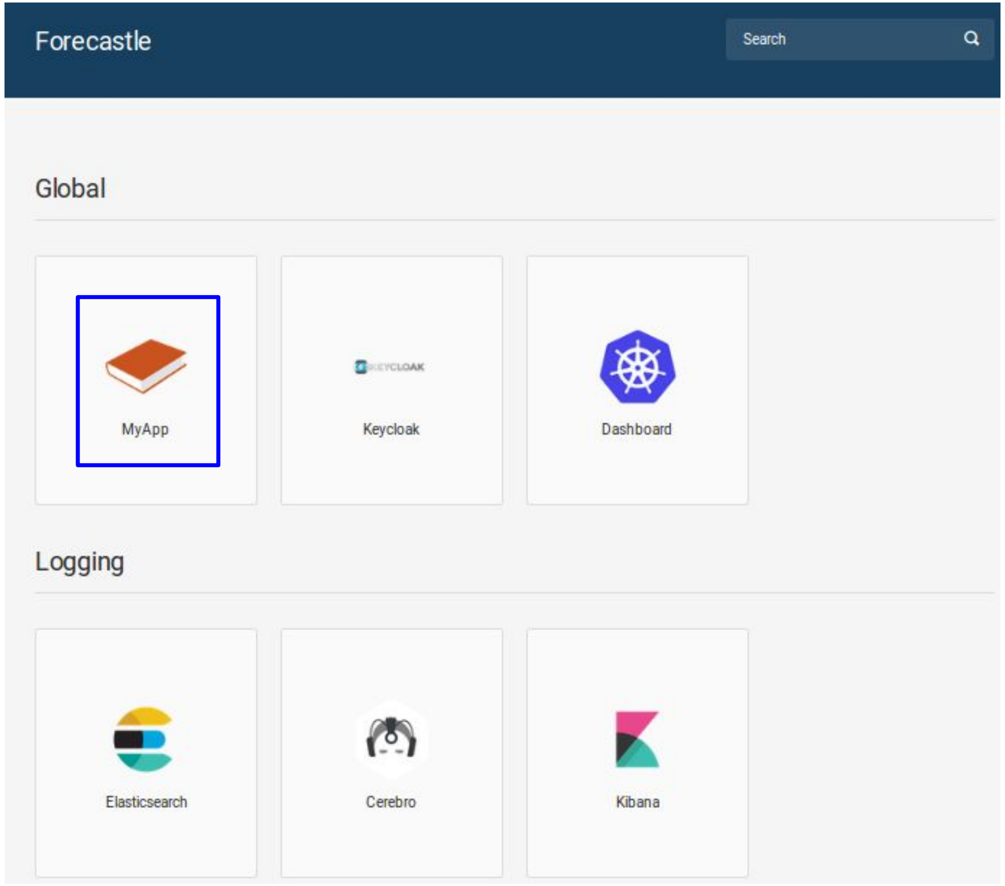
https://github.com/stakater/Forecastle

Stakater

What is Forecastle

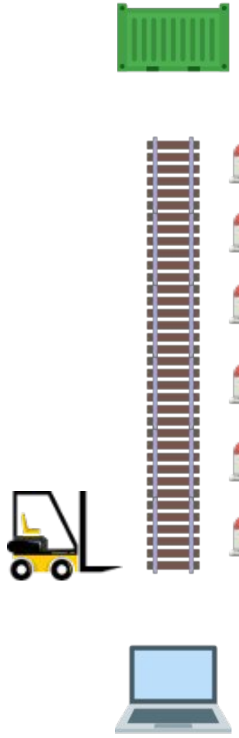
- ❖ Dashboard web page for services
- ❖ Automatically register apps based on Ingress

```
apiVersion: extensions/v1beta1,
kind: Ingress,
metadata:
  name: myapp-ingress
  annotations:
    forecastle.stakater.com/expose: true
    forecastle.stakater.com/appName: "MyApp"
...
```



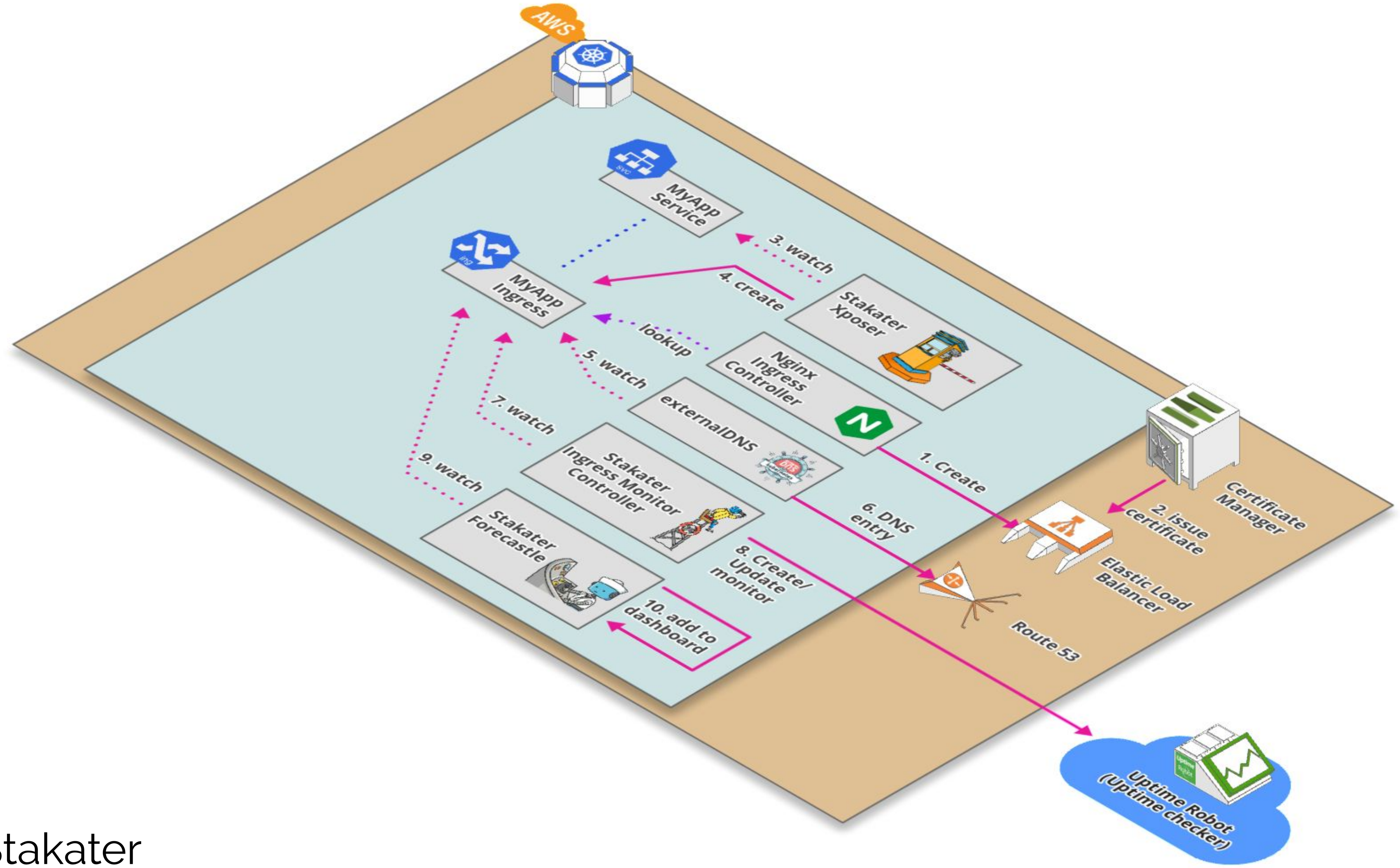
Stakater

Checkpoint



Stakater

Connecting the pieces


AWS
MyApp Service
MyApp Ingress
3. watch
4. create
Stakater Xposer
lookup
Nginx Ingress Controller
5. watch
externalDNS
7. watch
Stakater Ingress Monitor Controller
9. watch
Stakater Forecastle
10. add to dashboard
8. Create/ Update monitor
6. DNS entry
1. Create
2. issue certificate
Certificate Manager
Elastic Load Balancer
Route 53
Uptime Robot (Uptime checker)

Stakater

Recap

Manual approach

1. Create Service
2. Create Ingress
3. Create DNS record
4. Create TLS Certificate
5. Create Uptime Monitor
6. Bookmark Service URL

Efficient approach

Create Service

- → Ingress auto-generated
- → DNS record auto-generated
- → TLS Certificate auto-generated
- → Uptime Monitor auto-generated
- → Service auto-bookmarked

Thank you

Stakater